INFOPUC
INSTITUTO DE INFORMÁTICA

Manejo de hojas de Cálculo con **Excel Avanzado: Macros**

Alexander Labajos Trigoso — Desarrollo del libro
Erick Oscátegui Torres — Actualización
Elizabeth Gil Núñez — Revisión pedagógica
Úrsula León Castillo — Corrección de estilo
Sara Bravo Montenegro — Corrección de estilo

Avenida Universitaria 1801, Lima 32
Teléfono: (511) 626-2000/ anexo 3763 - 2603
Telefax: (511) 626-2885
Correo electrónico: infopuc@pucp.edu.pe
Página web: http://infopuc.pucp.edu.pe/

Este material ha sido elaborado por InfoPUC y es entregado a la Institución Educativa para su posterior distribución de manera gratuita a sus alumnos, como parte del contrato de servicios que han celebrado ambas instituciones. InfoPUC no se hace responsable frente a terceros por el uso que se realice respecto del presente material.

La información puesta a disposición a través de las referencias bibliográficas (páginas electrónicas, *blogs*, videos y audios) y todo material digital externo al presente libro pueden sufrir variaciones en el tiempo. El InfoPUC no asume ningún tipo de responsabilidad por la disponibilidad de las fuentes, ni por las modificaciones que la información haya podido sufrir.

Las imágenes utilizadas con fines educativos en los módulos de la presente publicación fueron tomadas de los *softwares* Microsoft Windows XP, Microsoft Office de titularidad de Microsoft Corporation.

Las marcas registradas son propiedad de sus respectivas compañías.

Esta publicación ha sido producida empleando Microsoft Office Word.

TABLA DE CONTENIDO

CAPÍTULO 1: CONCEPTOS INICIALES DE PROGRAMACIÓN EN EXCEL MACROS 9

1.1 ¿Qué es una macro? 12
1.2 Planificando las actividades para la grabación en macros 12
1.3 La realización del proceso de grabación de macros 13
1.4 Ejecutando una macro 15
1.5 Control de referencia en una grabación 15
1.6 Activando las herramientas de Visual Basic 17
1.7 Asignando un botón a una macro 18
1.8 Configuración de seguridad en las macros 19
1.9 Visual Basic para aplicaciones - VBA 20
1.10 El editor de VBA 22
1.10.1 Barra de menú 23
1.10.2 Barra de herramientas 24
1.10.3 Explorador de proyectos (ventana de objetos) 24
1.10.4 El formulario 25
1.10.5 La caja de controles 25
1.10.7 Área de programación 27
1.11 Primer programa utilizando Excel y VBA 28
1.12 Guardando un archivo con macros 30

CAPÍTULO 2: FUNDAMENTOS DE PROGRAMACIÓN 35

2.1 Conceptos iniciales de algoritmos 38
2.1.1 Concepto de algoritmo 39
2.1.2 Analizando el problema 39
2.2 El programa y sus partes 43
2.2.1 Cabecera del programa 43
2.3 Tipos de datos 45
2.3.1 Datos numéricos 45
2.3.2 Datos alfanuméricos 46
2.3.3 Datos fecha/hora 47
2.3.4 Datos lógicos 47
2.3.5 Declarando variables 48
2.3.6 Asignando datos a las variables 49
2.4 Tipos de operaciones en VBA 49

2.4.1 Operaciones aritméticas 50
2.4.2 Operaciones lógicas 50
2.4.3 Operaciones de concatenación 51
2.5 Desarrollando el programa 51
2.6 Trabajando con formularios 53
2.6.1 Etiquetas 53
2.6.2 Cajas de texto 53
2.6.3 Botones de comando 53

CAPÍTULO 3: PROGRAMACIÓN MODULAR 59
3.1 Diagrama de módulos 62
3.2 Subprogramas en VBA 67
3.2.1 Función 68
3.2.2 Invocando a una función 69
3.2.3 Procedimiento 70
3.2.4 Invocando a un procedimiento 70
3.3 Funciones predefinidas en VBA 72
3.3.1 Funciones de cadena de texto 72
3.3.2 Funciones matemáticas 74
3.3.3. Funciones de fecha y hora 76

CAPÍTULO 4: ESTRUCTURAS DE CONTROL 83
4.1 Estructuras de control condicional 87
4.1.1 Condicional simple de la forma: *If…then* 87
4.1.2 Condicional con contingencia de la forma: *If…then…else* 89
4.1.3 Condicionales anidadas 92
4.1.4 Condicional múltiple de la forma *Select…case* 95
4.2 Estructuras de control de ciclos repetitivos 102
4.2.1 Ciclos de la forma *For…Next* 102
4.2.2 Ciclos de la forma *Do….Loop While* 104

CUADRO DE CAPACIDADES:

NOMBRE DE LA UNIDAD – 1	CONTENIDOS	RECURSOS	CAPACIDADES ESPECÍFICAS				ACTITUDES Y VALORES	DURACIÓN
			Comprensión e información	Indagación y experimentación	Juicio crítico	Creatividad		
Conceptos iniciales de programación en Excel Macros	**Capítulo 1** 1.1 ¿Qué es una macro? 1.2 Planificando las actividades para la grabación en macros 1.3 La realización del proceso de grabación de macros 1.4 Ejecutando una macro 1.5 Control de referencia en una macro 1.6 Activando las herramientas de Visual Basic 1.7 Asignando un botón a una macro 1.8 Configurando la seguridad en las macros 1.9 Visual Basic para aplicaciones -VBA 1.10 El editor de VBA 1.11 Primer programa utilizando Excel y VBA 1.12 Guardando un archivo con macros Revisando lo aprendido **Proyecto integrador**: "Elaborando una calculadora con güincha: Diseñando el formulario de trabajo"	Computadora Proyector Office 2010 Manual	• Comprende los conceptos de grabación de macros mediante los ejercicios propuestos. • Identifica y describe las herramientas de Visual Basic. • Domina la terminología de Visual Basic. • Planifica, graba y ejecuta una macro. • Conoce el entorno del editor de VBA para la elaboración de programas de mayor complejidad. • Crea su primer programa utilizando el entorno de VBA.	• Aplica en las diversas actividades propuestas una serie de instrucciones propias del lenguaje de programación.	• Identifica la importancia de la aplicación Visual Basic. • Aprecia el uso de las macros para la automatización de tareas.	• Aplica las nuevas herramientas de macros a sus trabajos en Excel.	• Muestra actitud participativa en la clase. • Desarrolla responsable-mente los ejercicios. • Aplica la tecnología como recurso para mejorar la calidad de vida.	4 semanas

NOMBRE DE LA UNIDAD – 2	CONTENIDOS	RECURSOS	CAPACIDADES ESPECÍFICAS				ACTITUDES Y VALORES	DURACIÓN
			Comprensión e información	Indagación y experimentación	Juicio crítico	Creatividad		
Fundamentos de programación	**Capítulo 2** 2.1 Conceptos iniciales de algoritmos 2.2 El programa y sus partes 2.3 Tipos de datos 2.4 Variables en VBA 2.5 Tipos de operaciones en VBA 2.6 Desarrollando el programa 2.7 Trabajando con formularios Revisando lo aprendido **<u>Proyecto integrador</u>:** "Mejorando la presentación de tu formulario y analizando el problema a resolver"	Computadora Proyector Office 2010 Manual	• Discrimina los conceptos de algoritmos y diagramas de flujo reconociendo sus características propias. • Identifica las herramientas básicas para elaborar los algoritmos. • Identifica las partes de un programa. • Reconoce y comprende los pasos para desarrollar un programa. • Discrimina entre los diversos tipos de datos y tipos de operaciones que maneja el lenguaje VBA. • Define, identifica y aprende a trabajar con variables, reconociendo su importancia de uso en un programa.	• Descubre el uso de las nuevas herramientas a través de las actividades propuestas. • Explora los distintos elementos del diseño de pantalla de usuario. • Manipula y trabaja con formularios. • Selecciona el tipo de dato adecuado dependiendo de la variable que va a utilizar.	• Valora la importancia de realizar un algoritmo para elaborar un programa. • Aprecia la variedad de controles que ofrece el lenguaje de programación para lograr que el formulario sea más sencillo. • Valora la importancia del uso de variables y tipos de datos en la programación.	• Aplica las nuevas herramientas a sus trabajos en Excel. • Diseña sus propios programas.	• Muestra actitud participativa en la clase. • Desarrolla responsable-mente los ejercicios. • Valora las herramientas que ofrece Excel 2010 para la presentación de la información.	4 semanas

NOMBRE DE LA UNIDAD - 3	CONTENIDOS	RECURSOS	CAPACIDADES ESPECÍFICAS				ACTITUDES Y VALORES	DURACIÓN
			Comprensión e información	Indagación y experimentación	Juicio crítico	Creatividad		
Programación modular	**Capítulo 3** 3.1 Diagrama de módulos 3.2 Subprogramas en VBA 3.2.1 Función 3.2.2 Invocando a una función 3.2.3 Procedimiento 3.2.4 Invocando a un procedimiento 3.3 Funciones predefinidas en VBA 3.3.1 Funciones de cadena de texto 3.3.2 Funciones matemáticas 3.3.3 Funciones de fecha y hora Revisando lo aprendido **<u>Proyecto integrador</u>**: "Insertando código fuente al programa"	Computadora Proyector Office 2010 Manual	• Discrimina entre módulos y subprogramas. • Reconoce funciones y procedimientos. • Comprende el uso de invocar a funciones y a procedimientos. • Identifica argumentos válidos. • Identifica las características de cada una de las funciones predefinidas en VBA reconociendo en qué casos utilizarlas.	• Descubre el uso de las nuevas herramientas a través de las actividades propuestas.	• Identifica la importancia de la aplicación de funciones y procedimientos. • Aprecia el uso de los diagramas de módulos en la programación.	• Determina y crea sus propios diagramas de módulos. • Crea fórmulas con más de un argumento. • Crea su primer programa de acuerdo a los fundamentos de programación desarrollados.	• Muestra actitud participativa en la clase. • Desarrolla responsablemente los ejercicios. • Aplica la tecnología como recurso para mejorar la calidad de vida.	4 semanas

NOMBRE DE LA UNIDAD – 4	CONTENIDOS	RECURSOS	CAPACIDADES ESPECÍFICAS				ACTITUDES Y VALORES	DURACIÓN
			Comprensión e información	Indagación y experimentación	Juicio crítico	Creatividad		
Estructuras de control	**Capítulo 4** 4.1 Estructuras de control condicional 4.1.1 Condicional simple de la forma: *If…then* 4.1.2 Condicional con contingencia de la forma: *If…then...else* 4.1.3 Condicionales anidadas 4.1.4 Condicional múltiple d la forma: *Select…case* 4.2 Estructuras de control de ciclos repetitivos 4.2.1 Ciclos de la forma: *For…Next* 4.2.2 Ciclos de la forma: *Do…Loop While* Revisando lo aprendido **Proyecto integrador**: "Implementando las operaciones finales de la calculadora"	Computadora Proyector Office 2010 Manual	• Reconoce, comprende y aprende a diferenciar las estructuras de control condicional. • Reconoce, comprende y aprende a diferenciar las estructuras de control de ciclos repetitivos.	• Descubre, a través de las actividades propuestas, en qué casos usar estructuras de control y cuál de ellas utilizar según el caso o problema planteado.	• Evalúa la importancia de insertar una estructura de control óptimo para el problema.	• Aplica las nuevas herramientas a sus trabajos en Excel. • Crea su propio programa utilizando los conceptos de estructuras de control para optimizar los resultados.	• Muestra actitud participativa en la clase. • Desarrolla responsable-mente los ejercicios. • Aplica la tecnología como recurso para mejorar la calidad de vida.	4 semanas

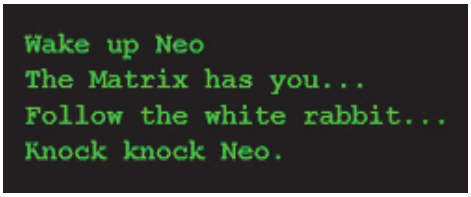
CAPÍTULO 1

CONCEPTOS INICIALES DE PROGRAMACIÓN EN EXCEL MACROS

Tomado de: <http://commons.wikimedia.org/wiki/File:Teachers_of_KMJH_2007-12-11.jpg>

Ana Luisa, profesora de Lengua de secundaria, tiene a cargo 10 salones de 25 alumnos cada uno. Es fin de bimestre, ella está sacando promedios por más de 4 horas y aún no termina. Debe registrar notas y, con la ayuda de una calculadora, obtener el promedio de cada alumno, luego, debe ver cuáles son los alumnos que obtuvieron el mejor puntaje por sección y, finalmente, hacer la relación de los 3 mejores por grado. Esta actividad le demanda demasiado tiempo, es aburrida y mecánica, pero implica concentración y debe estar muy pendiente de no cometer errores. ¿Cómo podemos aliviar el trabajo de Ana Luisa para que pueda invertir menos tiempo en esta actividad y dedicarlo a la planificación de otras actividades escolares?

Antes de empezar con el desarrollo del capítulo, te invitamos a que leas el siguiente artículo sobre una de las películas más famosas de los hermanos Wachowski: *The Matrix.*



Al ver este mensaje en su ordenador, Thomas A. Anderson (Keanu Reeves), programador informático y *hacker* que usa el alias Neo, queda totalmente intrigado: ¿qué es Matrix? y ¿por qué me posee? A partir de ese momento Neo confirma la hipótesis de que existe algo más y es Morfeo (Laurence Fishburne) quien lo conducirá a la respuesta que tanto ansía conocer.

Neo toma la pastilla roja que le ofrece Morfeo, la cual implica olvidar su vida actual y descubrir que el mundo en el que creía vivir no es más que una simulación virtual a la que se encuentra conectado mediante un cable enchufado en su cerebro. Los miles de millones de personas que viven (conectadas) a su alrededor están siendo cultivadas del mismo modo para poder dar energía a las máquinas. Esta ilusión colectiva (o simulación interactiva) es conocida como the Matrix,la matriz.

Morfeo le explica en qué consiste la realidad: se encuentran cerca del año 2199 y la humanidad está esclavizada por las máquinas, que tras el desarrollo de la inteligencia artificial se rebelaron contra su creador: el hombre. Las máquinas, tras vencer la guerra y quedar privadas de la energía solar que necesitaban para funcionar, ahora dominan la superficie terrestre y emplean a la especie humana como fuente de energía, cosechándolos en grandes campos de cultivo.

El mundo virtual de Matrix se convierte en el campo de batalla donde Neo tendrá que combatir contra sus agentes, que son unos programas en forma humana capaces de poseer a cualquier habitante de Matrix que la resistencia no haya liberado, y en particular contra el temible agente Smith (Hugo Weaving). Estos intentan impedir que los rebeldes rescaten a las personas que están conectadas, ya que están programados para capturar a cualquier intruso que altere la realidad de Matrix. En este mundo virtual, los seres humanos que son conscientes de la verdadera esencia de lo que les rodea, son capaces de desafiar parcialmente las leyes físicas y realizar hazañas asombrosas.

En la escena final, Neo está de vuelta a Matrix y habla por teléfono (sabiendo que la esta interviene la llamada), para decirle a las máquinas que ahora las cosas van a cambiar y que va enseñar a la gente "lo que no quieren que vean".

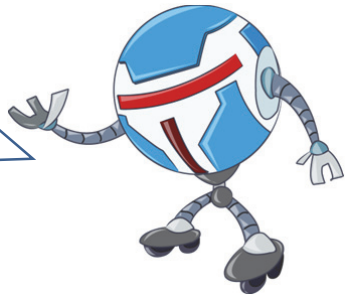
Tomado de: "The Matrix". *Wikipedia, la enciclopedia libre*. Fecha de consulta: 30/10/2011 <http://es.wikipedia.org/wiki/The_Matrix>

Ahora bien, así como la Matrix a través de un sistema de códigos y programas controla y hace posible que exista un mundo ficticio, las macros en Excel son generadas a través de un lenguaje de programación llamado Visual Basic para Aplicaciones (VBA).

¿Sabías que…?

VBA es una versión modificada del lenguaje de programación Visual Basic que trabaja de forma integrada con el libro de trabajo de Excel.

En este capítulo, aprenderás más acerca de este lenguaje, asimismo, crearás y ejecutarás tus propias macros.



Tu trabajo

A continuación, te presentamos un pantallazo del primer producto que elaborarás con Excel macros, el cual te permitirá calcular los promedios rápidamente, solo tendrás que ingresar los datos y automáticamente la macro te mostrará el promedio final y las personas que han obtenido los 3 primeros puestos.

	A	B	C	D	E	F	G	H	I	J
1	Concurso de Matemática									
2	Etapa Final									
3										
4	Nro.	Participante	Sección	Prueba 1	Prueba 2	Prueba 3	Promedio			
5	1	Milagros Córdova	B	18	19	16	17.7			
6	2	Walter Torres	D	17	14	20	17.0			
7	1	Marco Labajos	C	17	14	18	16.3			
8	6	Juan Manuel Rojas	C	16	14	18	16.0			
9	4	Frank Gutiérrez	G	14	17	16	15.7			
10	3	Sofía Díaz	A	16	16	15	15.7			
11	5	Rosario Contreras	E	12	20	15	15.7			
12	7	Carlos Roca	C	10	13	20	14.3			
13	9	Arturo Villa	B	11	11	13	11.7			
14	8	Adriana Céspedes	A	12	10	12	11.3			

Calcular

Mostrar

Resultado

Primer Puesto: Milagros Córdova, nota=17.7

OK

En este capítulo, tu trabajo consistirá en elaborar la tabla de puntuaciones y premiación de los ganadores de la etapa final del concurso de matemática de tu colegio.

Antes de empezar con el desarrollo del capítulo, debes recordar qué es una macro.

1.1 ¿Qué es una macro?

Una macro es una herramienta que permite automatizar varias tareas cotidianas, es decir, el usuario podrá evitar la ejecución repetitiva de estas tareas porque solo tendrá que ejecutar esta para poder realizarlas de forma automática.

Considera la grabación de una macro cuando observes que pulsas las mismas teclas y realizas una secuencia de opciones en forma repetitiva.

Hazlo tú mismo:

Escribe algunos ejemplos en los cuales podrías aplicar macros. Discútelo con tus compañeros de clase:

_____
_____
_____
_____
_____

1.2 Planificando las actividades para la grabación en macros

Antes de iniciar la grabación de una macro, debes planificar las actividades a realizar, tales como la selección de hojas, celdas, herramientas u objetos que formarán parte de esta.

Hazlo tú mismo:

Descarga de la plataforma el archivo **puntuaciones.xlsx**.

	A	B	C	D	E	F	G
1	Concurso de Matemática						
2	Etapa Final						
3							
4	Nro.	Participante	Sección	Prueba 1	Prueba 2	Prueba 3	Promedio
5	1						
6	2						
7	3						
8	4						
9	5						
10	6						
11	7						
12	8						
13	9						
14	10						

Ingresa los nombres de los participantes de la competencia, así como su sección y las notas obtenidas en cada prueba.

Con esta información, se desea generar una macro que haga lo siguiente:

1) Calcule el puntaje promedio de cada uno de los participantes. Este puntaje debe ser redondeado a un decimal.

2) Ordene los puntajes promedio de mayor a menor.

3) Cambie el color de fuente y relleno a las celdas que contienen los puntajes de los 3 primeros lugares del concurso de la siguiente manera:

Puesto	Color de relleno	Color de fuente
1	Amarillo	Rojo
2	Canela fondo 2	Verde
3	Púrpura énfasis 4 claro 60%	Azul oscuro texto 2

A continuación, escribe la secuencia de pasos que vas a realizar para generar esta macro:

Paso 1: Ubícate en la celda G5

Paso 2:_____

Paso 3:_____

Paso 4:_____

Paso 5:_____

Paso 6:_____

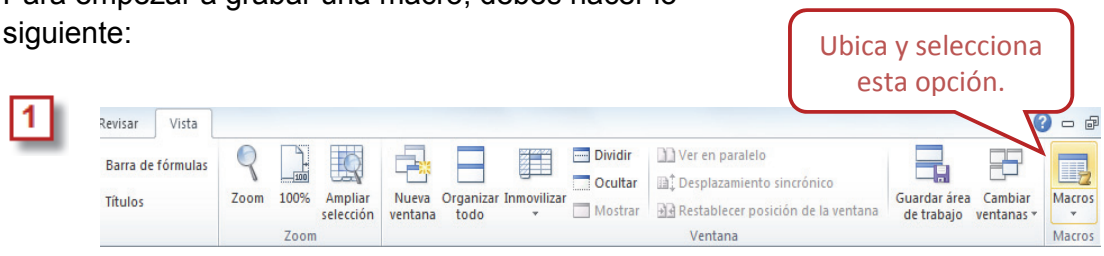
Paso 7:_____

Paso 8:_____

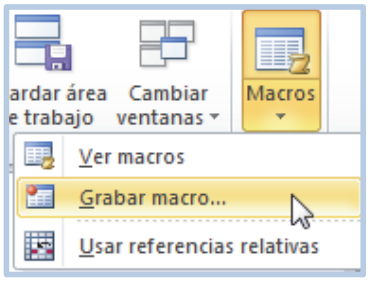
1.3 La realización del proceso de grabación de macros

La grabadora de macros almacena acciones que el usuario realiza o comandos que elige mientras trabaja.

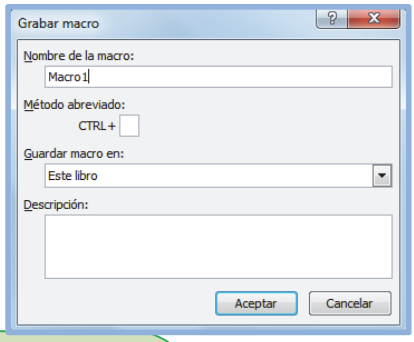
Para empezar a grabar una macro, debes hacer lo siguiente:



2 Luego, haz clic en la lista del botón **Macros** y elige **Grabar macro**.



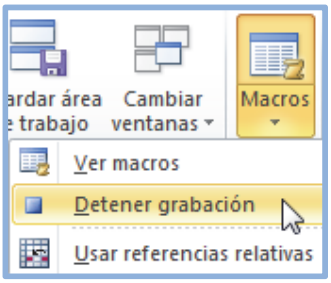
3 Para darle a la macro un nombre distinto del que sugiere Excel, digita un nombre en el cuadro **Nombre de la macro**.



¡Importante!

- Puedes asignar a la macro un método abreviado de teclado. Los métodos abreviados pueden ser **CTRL+ cualquier letra** o **CTRL+ MAYÚS+ cualquier letra**. Recuerda que estos reemplazarán a los métodos abreviados de Excel mientras esté abierto el libro que contiene la macro.
- Para que una macro esté disponible todo el tiempo, pulsa el ícono **Grabar macro**; después, en la opción **Guardar macro en**, elige la alternativa **Libro de macros personal**, de esta forma el libro estará siempre abierto.

4 Realiza las acciones que se van a grabar y, cuando hayas terminado, haz clic en la lista del botón **Macros** y elige **Detener grabación**.



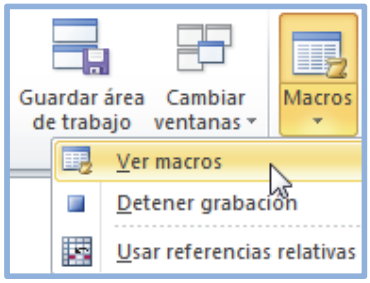
Hazlo tú mismo:

Utilizando el archivo **puntuaciones.xlsx**, aplica los pasos que habías definido en la etapa de planificación y graba la macro de nombre **Calcula_Ganador**. Asigna a esta macro un método abreviado con las teclas **CTRL+Mayús+J** y gúardalo en el libro de macros personal.

1.4 Ejecutando una macro

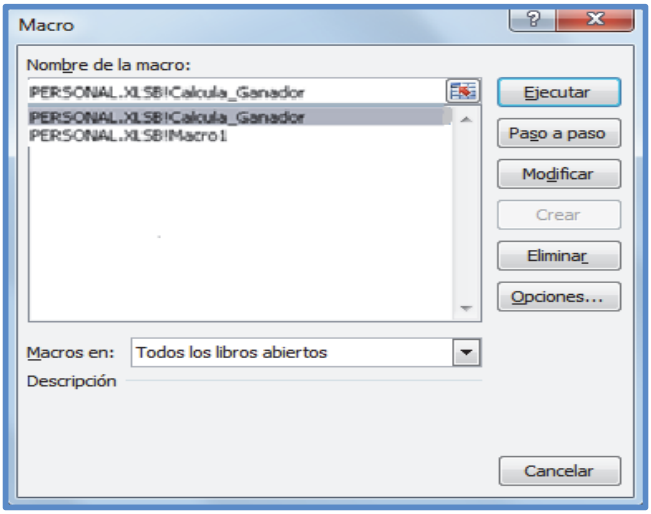
Cuando se ejecuta una macro, la secuencia de instrucciones grabadas indica a Excel lo que debe hacer. La grabadora de macros repite lo que el usuario realizó así como una grabadora de audio repite lo que una persona haya hablado.

Para ejecutar una macro, debes hacer lo siguiente:



1. Ubícate en la celda donde iniciarás la ejecución de la macro y, luego, selecciona la siguiente opción:

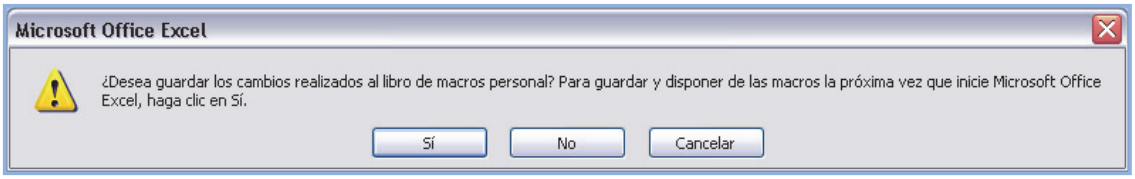
2. En la siguiente ventana, selecciona la macro que deseas ejecutar y haz clic en **Ejecutar**.



Hazlo tú mismo:

A continuación, realiza los siguientes pasos antes de ejecutar la macro **Calcula_Ganador**.

- Cierra el archivo **puntuaciones.xlsx** (no guardes los cambios).
- Cierra Excel y, cuando aparezca el siguiente mensaje, haz clic en la opción **Sí.**



Ahora sí, ejecuta la macro **Calcula_Ganador** con los pasos que has aprendido anteriormente.

1.5 Control de referencia en una grabación

Una referencia es la ubicación de una celda en Excel. Existen dos tipos de referencias: absoluta y relativa.

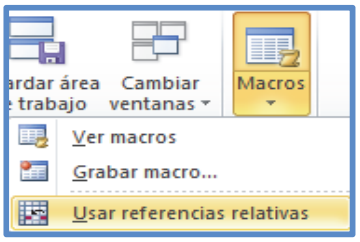
Para que puedas descubrir la diferencia entre la grabación de macros con referencias absolutas y relativas, te invitamos a realizar las siguientes acciones:

DESCUBRE Y APRENDE

1) Abre una hoja de Excel e ingresa los siguientes datos en las celdas indicadas:

	A	B	C	D
1		Prueba 1		Prueba 2
2	Letras	25		30
3	Ciencias	35		20
4				

2) Selecciona la celda B3 e inicia la grabación de la macro.
3) Selecciona la celda B4 y escribe la fórmula =B2+B3.
4) Detén la grabación y borra el contenido de la celda B4.
5) Selecciona D3 y ejecuta la macro.
   ¿En qué celda aparece el resultado?_____
   ¿Qué tipo de grabación crees que se está aplicando?_____
6) Borra el contenido de la celda B4.
7) Selecciona la celda B3 e inicia la grabación de una nueva macro.
8) Ubica y selecciona la siguiente opción:



9) Selecciona la celda B4 y escribe la fórmula =B2+B3.
10) Detén la grabación y borra el contenido de la celda B4.
11) Selecciona D3 y ejecuta la macro.
   ¿En qué celda aparece el resultado?_____
   ¿Qué tipo de grabación crees que se está aplicando?_____

En conclusión: Si guardas una macro grabada con referencias absolutas, Excel lleva un control de la posición exacta de cada celda seleccionada. En cambio, si grabas una macro con referencias relativas, Excel llevará un control de cada celda seleccionada en relación a la seleccionada anteriormente.

Ejercicio de aplicación 1

1) Crea una macro en la celda A1 y colócale como nombre **relleno**. Esta macro debe hacer lo siguiente:
   a) Desplazar el cursor hasta la celda A5
   b) Rellenar esta celda de color amarillo
2) Detén la macro.
3) Ubícate en la celda C2 y ejecuta la macro.

¿Cuál ha sido la celda afectada al ejecutar la macro?_____

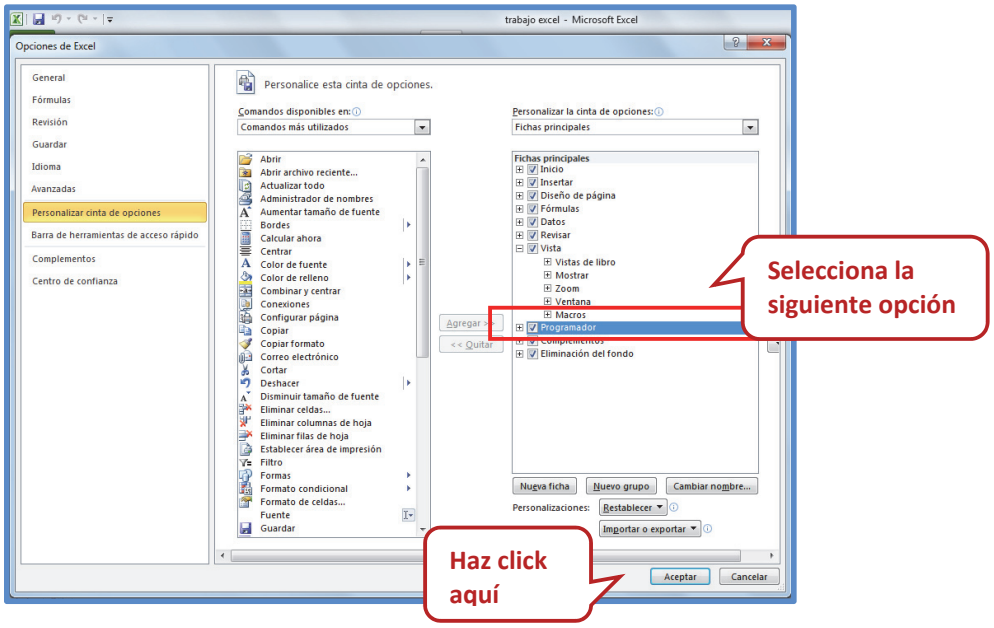
En la grabación de esta macro se está usando una referencia_____

4) Ahora, vas a crear una nueva macro, para ello ubícate en la celda E2 y selecciona la opción **usar referencias relativas**.
5) Luego, graba la macro y colócale como nombre "texto". Esta macro debe hacer lo siguiente:
   a) Desplazar el cursor hasta la celda E10
   b) Rellenar esta celda de color amarillo
6) Detén la macro.
7) Ubícate en la celda H7 y ejecuta la macro.

¿Cuál ha sido la celda afectada al ejecutar la macro?_____

1.6 Activando las herramientas de Visual Basic

Para la creación y manejo de macros avanzadas, Excel posee un conjunto de herramientas ubicadas en una ficha llamada **Programador**. Esta ficha no se encuentra disponible dentro de la instalación básica de Excel 2010. Para activar esta ficha, debes hacer lo siguiente:



DESCUBRE Y APRENDE

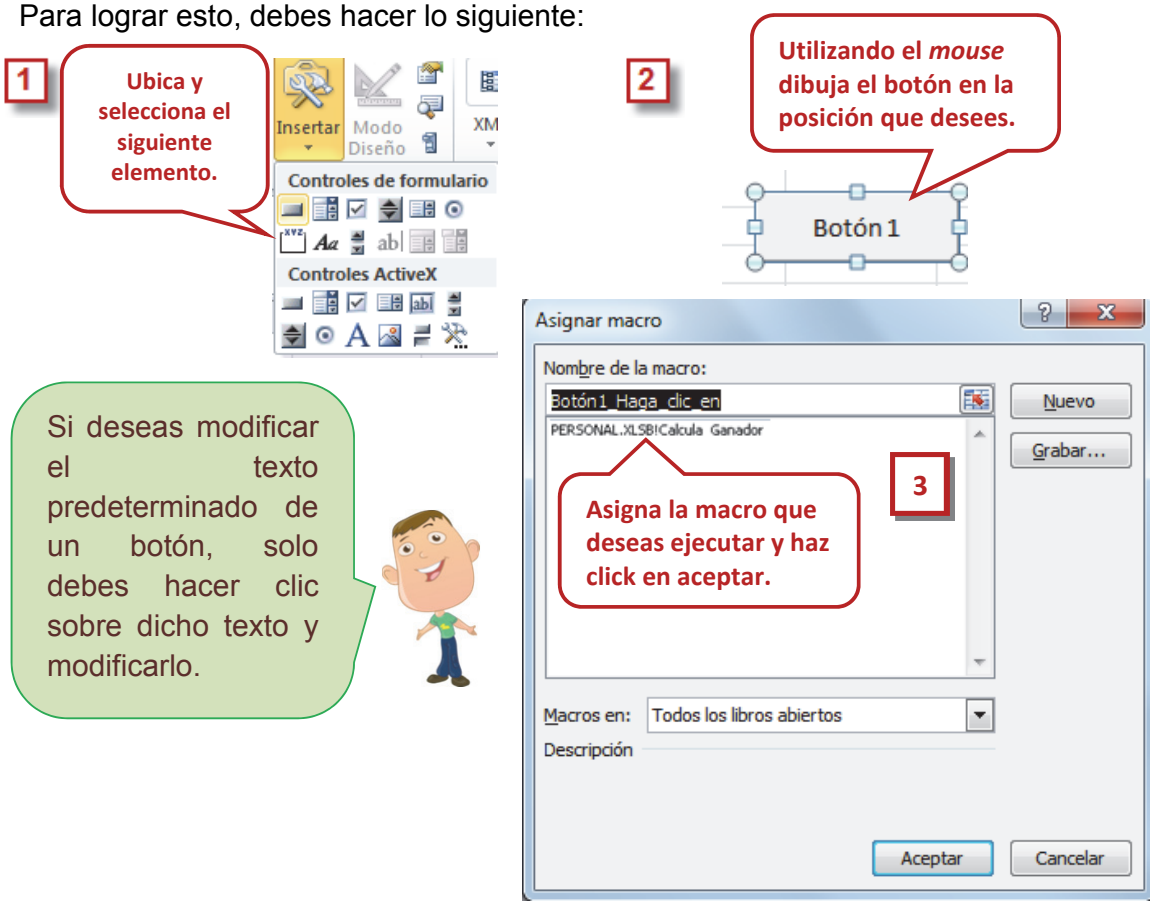
Explora las herramientas de Visual Basic y relaciona las columnas:

Elemento	Funcionalidad
	Usar referencias relativas
	Insertar controles
	Editor de Visual Basic
	Grabar macro
	Ver macros
	Seguridad de macros

1.7 Asignando un botón a una macro

Con la herramienta **Insertar controles**, es posible asignar un botón a una macro, para que se pueda ejecutar con solo hacer clic en este.

Para lograr esto, debes hacer lo siguiente:



Hazlo tú mismo:

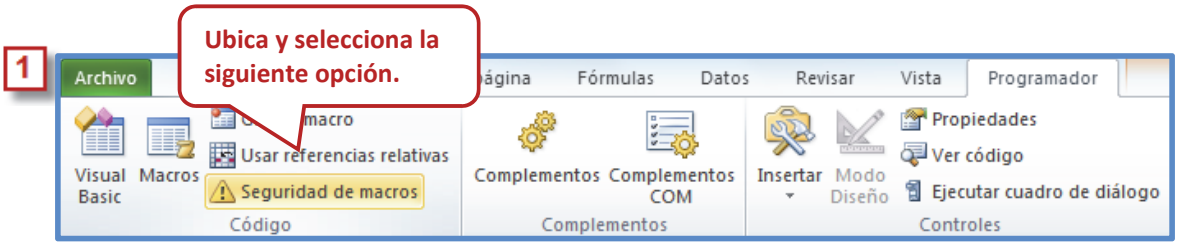
En el archivo **puntuaciones.xlsx**, asigna un botón a la macro **Calcula_Ganador** y prueba su ejecución. Coloca como texto del botón la palabra “Calcular”.

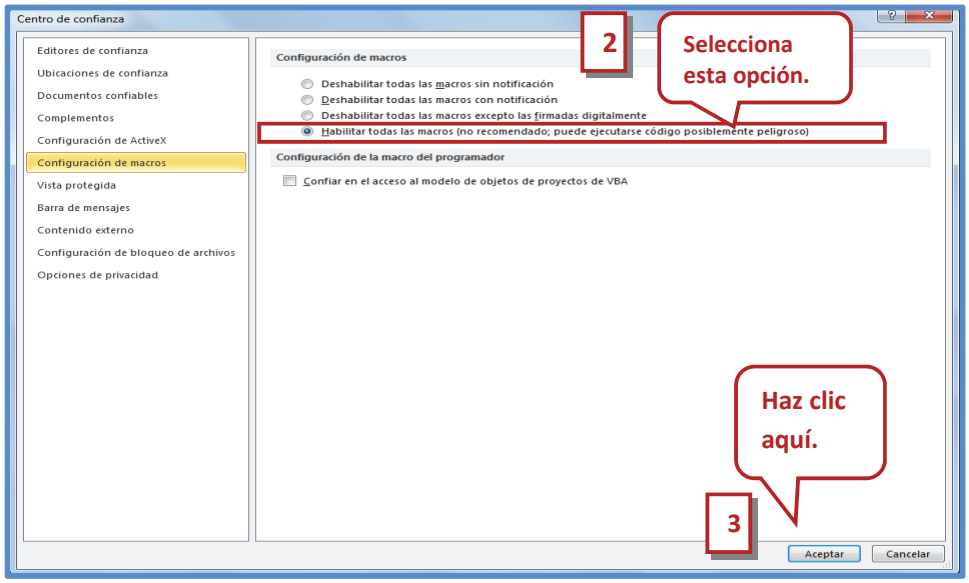
Hasta el momento tu trabajo debe quedar así:

	A	B	C	D	E	F	G	H
1	Concurso de Matemática							
2	Etapa Final							
3								
4	Nro.	Participante	Sección	Prueba 1	Prueba 2	Prueba 3	Promedio	
5	10	Milagros Córdova	B	18	19	16	17.7	Calcular
6	2	Walter Torres	D	17	14	20	17.0	
7	1	Marco Labajos	C	17	14	18	16.3	
8	6	Juan Manuel Rojas	C	16	14	18	16.0	
9	4	Frank Gutiérrez	G	14	17	16	15.7	
10	3	Sofía Díaz	A	16	16	15	15.7	
11	5	Rosario Contreras	E	12	20	15	15.7	
12	7	Carlos Roca	C	10	13	20	14.3	
13	9	Arturo Villa	B	11	11	13	11.7	
14	8	Adriana Céspedes	A	12	10	12	11.3	

1.8 Configuración de seguridad en las macros

En algunos casos, Excel 2010 está programado para que la seguridad en las macros sea alta, lo cual hace imposible que se puedan ejecutar. Si vas a ejecutar programas macro, deberás cambiar la configuración de seguridad; para ello, debes hacer lo siguiente:





1.9 Visual Basic para aplicaciones - VBA

Pertenece a una versión de Visual Basic que viene incluida en Excel. VBA es un lenguaje de programación estructurado con secuencias de comandos o lenguaje de macros. Una macro de VBA no puede ejecutarse independientemente del Excel.

Con ayuda de tus compañeros, averigua algunas tareas que puedes realizar a través de un programa desarrollado en VBA.

_____
_____
_____

Terminología de VBA

Al trabajar con Visual Basic para aplicaciones, será necesario que conozcas el significado de algunos términos de importancia: aplicación, formulario, tiempo de diseño, tiempo de ejecución, objetos, propiedades, métodos y eventos.

a. Aplicación

Es el programa que has desarrollando utilizando las herramientas de VBA.

b. Tiempo de diseño

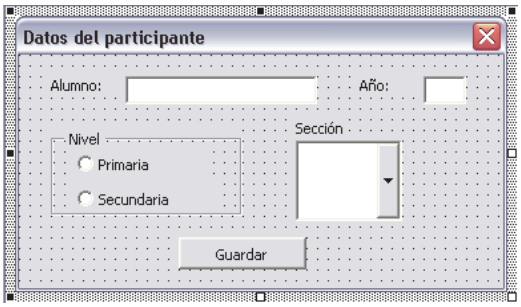
Es el momento en que se está diseñando la aplicación.

c. Tiempo de ejecución

Es el momento en que estás ejecutando la aplicación.

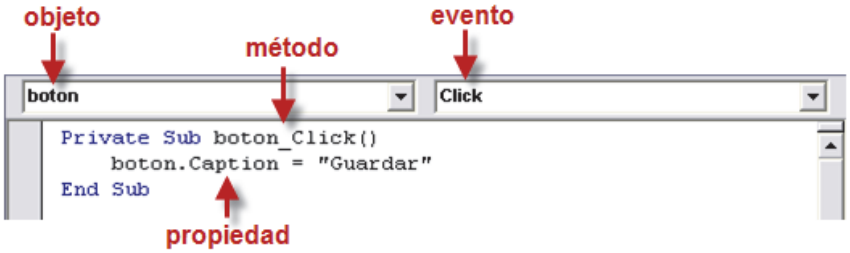
d. Formulario

Es el área donde se colocan los objetos gráficos para la aplicación. Un formulario está compuesto por controles.
Por ejemplo:



e. Objetos

Un objeto representa un elemento de una aplicación, por ejemplo, una hoja de cálculo, una celda, un formulario, un control, etcétera.



f. Propiedades

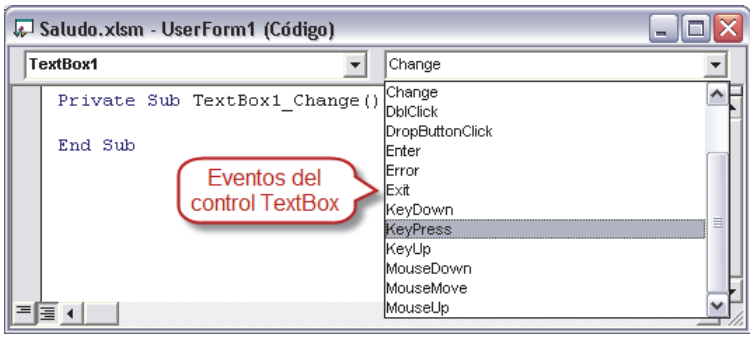
Son las características de un objeto, las cuales logran definir su comportamiento.

g. Métodos

Son las acciones que un objeto puede realizar. Un método es un procedimiento o función que se asocia a un tipo de objeto (botón, caja de texto, etcétera).

h. Eventos

Es una notificación de que algo ha ocurrido con el objeto. Son respuestas a acciones del usuario, por ejemplo, pulsar teclas, eventos del *mouse* (al hacer clic, doble clic, etcétera) o elegir opciones de un menú.



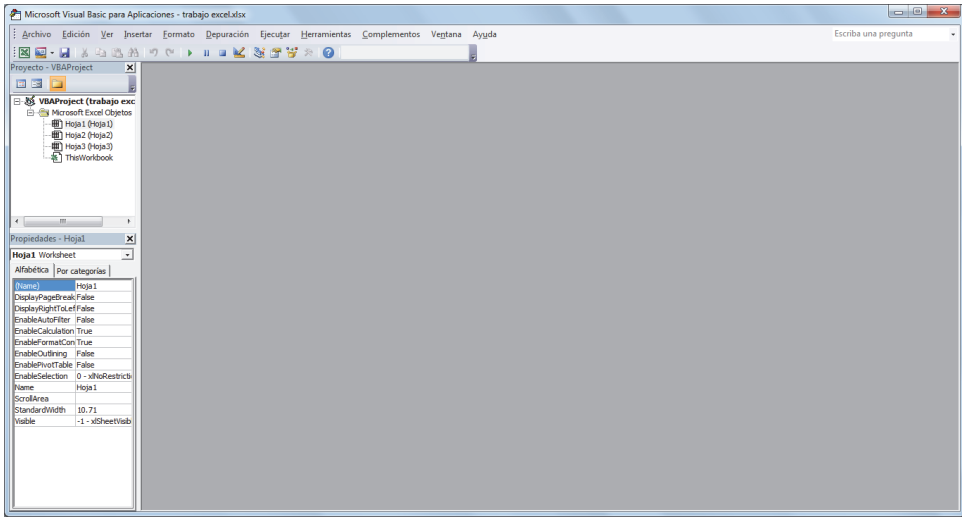
En VBA se pueden producir dos tipos de eventos: los eventos iniciados por el usuario y los iniciados por el sistema.

1.10 El editor de VBA

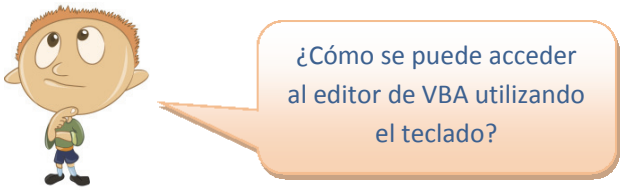
El editor de Visual Basic (VBE) contiene todas las herramientas de programación necesarias para escribir y editar programas que, luego, podrás ejecutar en tu libro de trabajo de Excel.

Para ingresar al editor de VBA, debes hacer clic en el ícono  que está ubicado dentro de las herramientas de Visual Basic.

Esta es la ventana que se presenta al ingresar al editor de VBA:

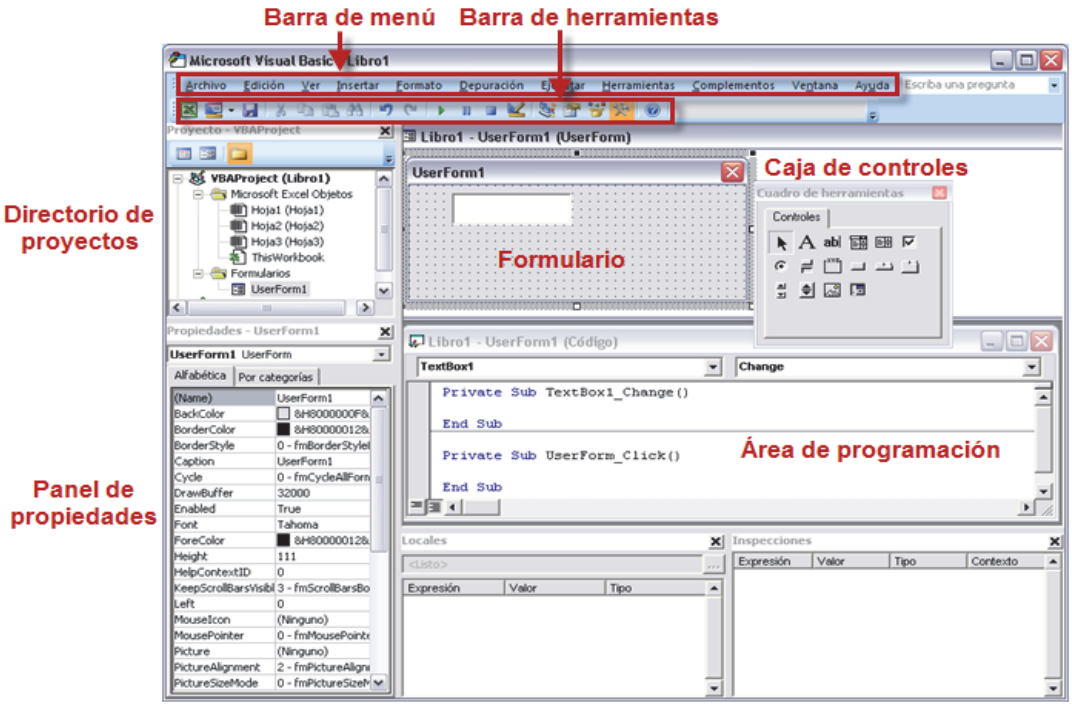


Averigua…



Responde:

Una vez dentro del editor, podrás agregar los elementos y utilizar las herramientas necesarias que te permitan desarrollar tu aplicación o programa en VBA.
Aquí puedes ver los principales elementos del editor de VBA:



1.10.1 Barra de menú

Proporciona las herramientas necesarias para desarrollar, probar y archivar la aplicación. A continuación, verás las principales opciones de la barra de menú:

- **Menú Archivo:** contiene las opciones que te permitirán administrar los archivos de trabajo de tu aplicación.
- **Menú Ver:** permite visualizar diversas ventanas del entorno.
- **Menú Insertar:** permite insertar los elementos necesarios para desarrollar tu aplicación, tales como un procedimiento, un formulario, un módulo, etcétera.
- **Menú Depuración:** aquí se encuentran todas las opciones que te ayudarán a verificar errores en tu programa.
- **Menú Ejecutar:** principalmente, se utiliza para ejecutar o detener la ejecución de un programa.

DESCUBRE Y APRENDE

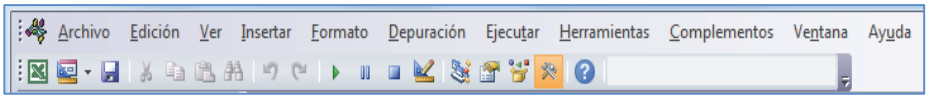
1) Explora las opciones de la barra de menú de VBA y completa la tabla.

Ícono	Descripción	Menú

2) Explora y completa el uso de las siguientes teclas o combinaciones de teclas dentro del editor de VBA:
   - F5:_____
   - F8:_____
   - CTRL+F8:_____
   - F2:_____
   - CTRL+Interrrumpir:_____

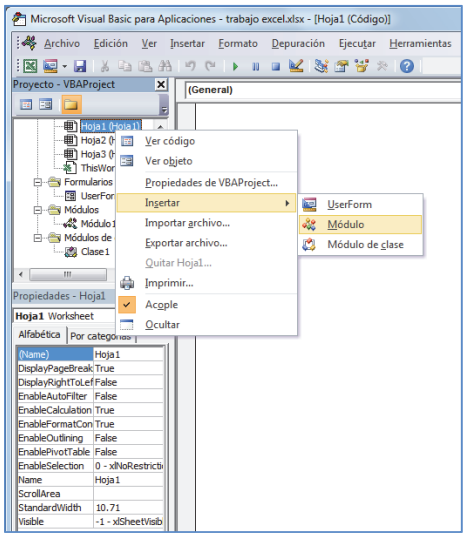
1.10.2 Barra de herramientas

Permite activar las tareas más comunes sin necesidad de utilizar los menús.
Es un conjunto de pequeños botones (íconos) situados debajo de la barra de menús. Su función es agilizar la elección de las opciones más utilizadas.

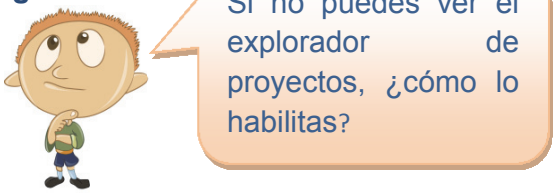


1.10.3 Explorador de proyectos (ventana de objetos)

Muestra un diagrama de árbol con los libros de trabajo actualmente abiertos en Excel, donde cada libro es un proyecto. Desde esta ventana se pueden añadir, seleccionar o eliminar los objetos del proyecto.



Averigua…



Responde:

_____

1.10.4 El formulario

Es la ventana donde se diseña la aplicación. El formulario es un contenedor en el que se colocan los controles (etiquetas, cuadros de texto, cajas de verificación, botones de opción, listas, etcétera). Contiene todos los elementos que se esperaría encontrar como parte de la ventana de un programa.

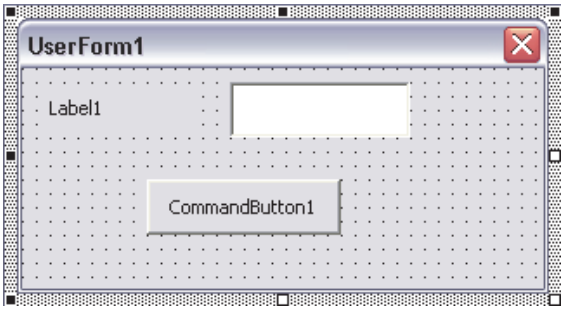
La cuadrícula de diseño permite alinear los controles fácilmente, al mismo tiempo que diseña su interfaz.

1.10.5 La caja de controles

Se utiliza para el diseño de la interfaz de la aplicación. En ella se encuentran las herramientas necesarias para el diseño y creación de los controles. Además, incluye un puntero para su manejo.

Ejercicio de aplicación 2

Abre una hoja de Excel, ingresa al editor de VBA y crea el siguiente formulario utilizando los elementos de la caja de controles.

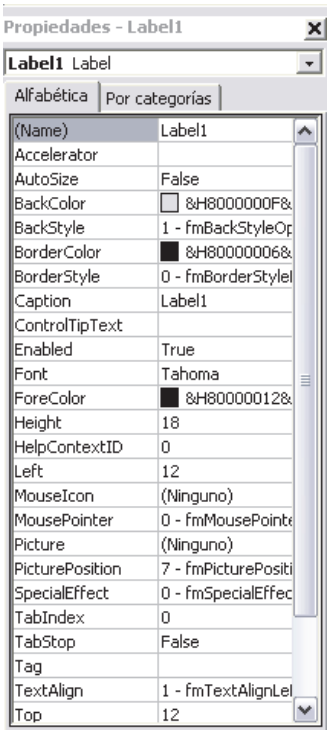


Escribe el nombre de los controles que utilizaste:

_____ _____ _____

1.10.6 La ventana de propiedades

Una propiedad es una característica de un objeto tal como su nombre o color. A través de esta ventana se configuran las propiedades de un objeto en tiempo de diseño.



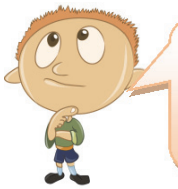
¡Importante!

- Se asigna un nombre predeterminado a cada uno de los controles existentes de un formulario. Por ejemplo, el primer control **Label** que se adicione al formulario se denominará **Label1**; si se agrega otro del mismo tipo, se denominará **Label2** y así sucesivamente.
- La propiedad **Name** o nombre, identifica un control. Cuando un formulario tiene muchos controles del mismo tipo, se puede utilizar la propiedad nombre del control para identificarlo.

A continuación, se coloca una lista con algunas de las propiedades más comunes de los formularios.

- **Backcolor:** define el color de fondo del objeto.
- **Name:** es el nombre del objeto.
- **Caption:** es el título del objeto.
- **Enabled:** si se coloca “False”, el objeto no responderá a ningún evento. Si el valor de la propiedad es “True”, responderá a sus eventos asociados.
- **Visible:** permite mostrar u ocultar un objeto (True/False).
- **Font:** es el formato de fuente del texto que posee el objeto.

Averigua…



¿Cómo habilitas la ventana de propiedades utilizando el teclado?

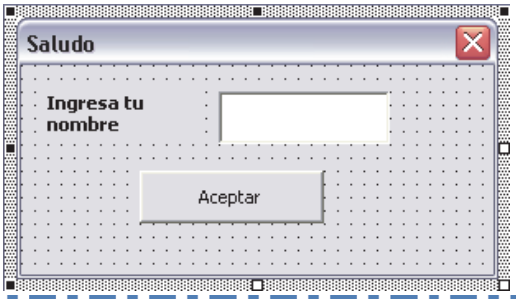
Responde:

Ejercicio de aplicación 3

Haz clic en cada uno de los controles que adicionaste al formulario del ejercicio anterior y modifica sus propiedades de acuerdo con lo que se indica en la siguiente tabla:

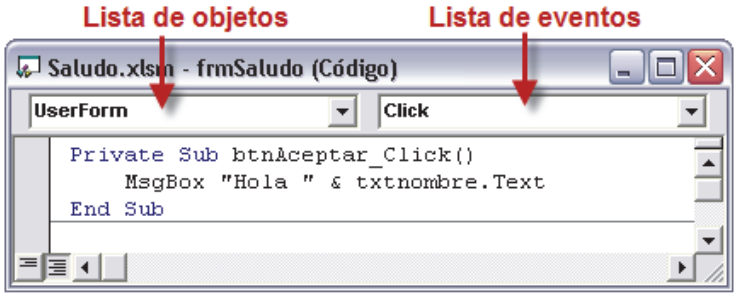
Control	Propiedad	Valor
UserForm1	Caption	Saludo
UserForm1	Name	frmSaludo
Label1	Name	lblNombre
Label1	Caption	Ingresa tu nombre
Label1	Font	Tahoma Bold 8
TextBox1	Name	txtNombre
CommandButton 1	Name	btnAceptar
CommandButton 1	Caption	Aceptar

Tu formulario debe quedar así:



1.10.7 Área de programación

Es el lugar donde se escribirá el código que se utilizará para responder a cada evento.



- **Lista de objetos:** es la lista de objetos del formulario (CommandButton, TextBox, CheckBox, etcétera).

- **Lista de eventos:** es la lista de eventos del objeto activo (Click, Dblclick, KeyPress, Change, MouseDown, etcétera).

Ejercicio de aplicación 4

1) Ingresa el siguiente código en la ventana de programación del formulario que utilizaste en el ejercicio anterior:

```
Private Sub btnAceptar_Click()
    MsgBox "Hola " & txtnombre.Text
End Sub
```

2) Ejecuta y prueba este programa.

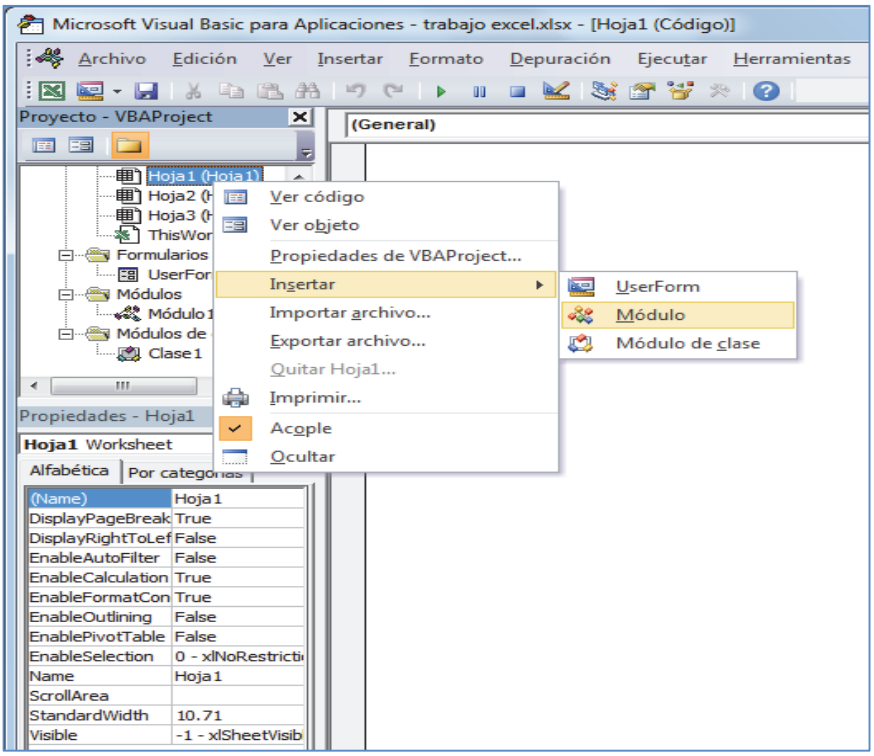
1.11 Primer programa utilizando Excel y VBA

Ahora verás cómo hacer un primer programa en VBA utilizando los datos de la hoja de cálculo Excel.

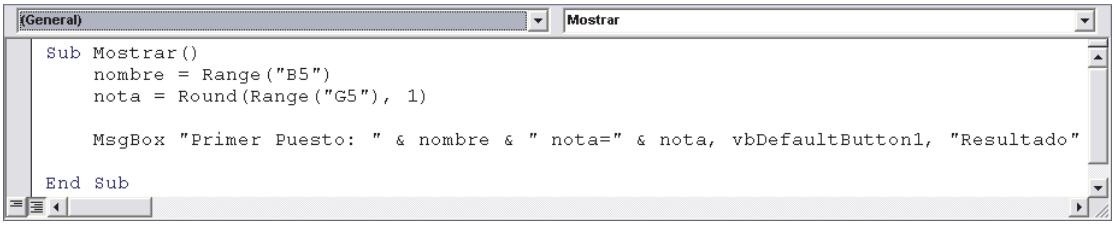
Este programa utilizará la hoja de cálculo **puntuaciones.xlsx**, que habías utilizado en la parte inicial de este capítulo.

Para ello, se te pide hacer lo siguiente:

1) Ingresa al editor de VBA.
2) En el explorador de proyectos, haz clic derecho en la Hoja1 e inserta un módulo.



3) Ingresa el siguiente código en el módulo que has creado.



4) Inserta un nuevo botón en la hoja de cálculo y colócale como texto la palabra "Mostrar".
5) Asocia a este botón el subprograma **Mostrar**, cuyo código ingresaste en el punto 3.
6) Haz clic en el botón **Calcular** para ejecutar la macro **Calcula_Ganador.**
7) Haz clic en el botón **Mostrar.**

Comenta el resultado obtenido:

_____

_____

Ahora analiza el código que has ingresado y contesta las siguientes preguntas:

- Con la instrucción: **nombre=Range("B5")**, ¿qué acción se está realizando?

  _____

- ¿Para qué sirve la instrucción MsgBox?

  _____

Ejercicio de aplicación 5

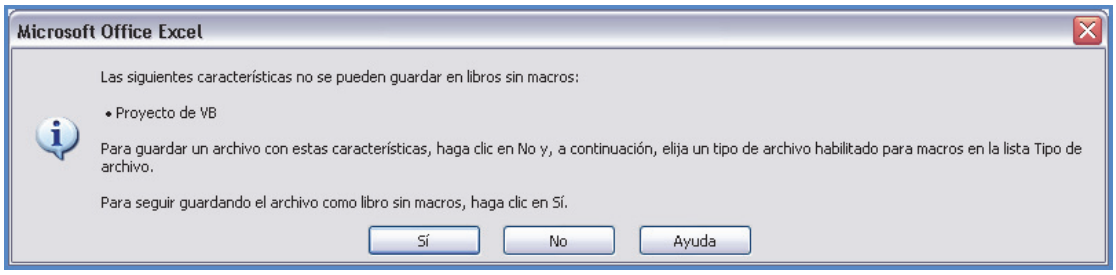
1) Crea un programa en VBA que permita convertir minutos a horas tal como se muestra en la siguiente figura:

Nota: Para convertir a número un dato ingresado como texto, se debe aplicar la función Val, así por ejemplo: Val(TextBox1.txt)

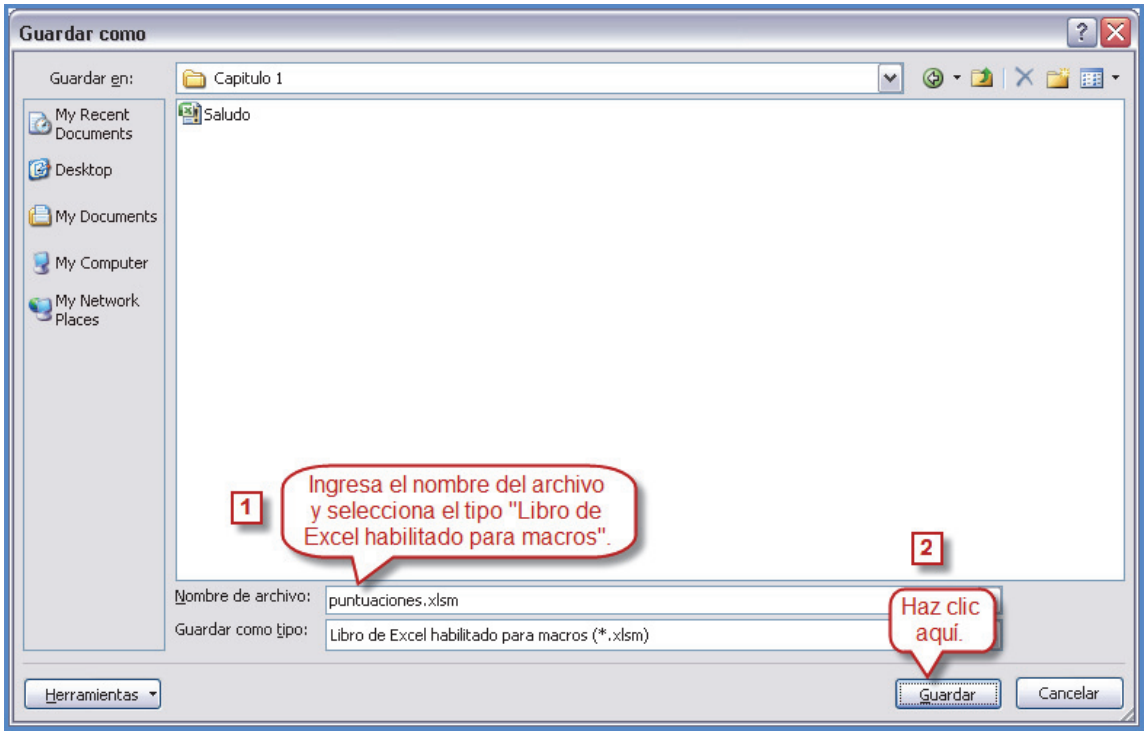
2) Ejecuta y prueba este programa.

1.12 Guardando un archivo con macros

Al intentar guardar un archivo que posee macros, aparecerá el siguiente mensaje:



Por tanto, debes seleccionar la opción **No,** luego de lo cual aparecerá la siguiente ventana:



Hazlo tú mismo:

Sigue los pasos anteriores y guarda el archivo **puntuaciones.xlsx** como libro de Excel habilitado para macros.

¿CUÁNTO APRENDÍ?

I. Marca V (verdadero) o F (falso), según corresponda:

1. El lenguaje de programación que se utiliza para macros en Excel se llama Visual Basic. ( )
2. La herramienta **Depurar paso a paso** sirve para ejecutar la macro instrucción por instrucción en el editor de VBA. ( )
3. Un evento puede ser generado por la pulsación de una tecla. ( )
4. Las macros se guardan con un tipo de archivo diferente al de una hoja de cálculo. ( )
5. No es posible detener una macro mientras se está ejecutando ( )

II. Abre una hoja de Excel y graba una macro de nombre **Texto** que realice lo siguiente:

1. Inserta en la celda A1 un texto.
2. Coloca este texto subrayado, negrita, cursiva y un tamaño de letra 24.
3. Aplica un fondo de color azul a la celda en la cual está el texto.
4. Añade el color del texto en rojo.
5. Añade un borde superior y un borde inferior a la celda.
6. Ajusta automáticamente la columna en la cual se ha insertado el texto.

Prueba la macro para comprobar que funciona.

Crea un botón en la barra de herramientas que al pulsarlo ejecute la macro anterior.

Prueba el botón que acabas de crear para comprobar que llama a la macro.

III. Relaciona los siguientes controles con su nombre correspondiente:

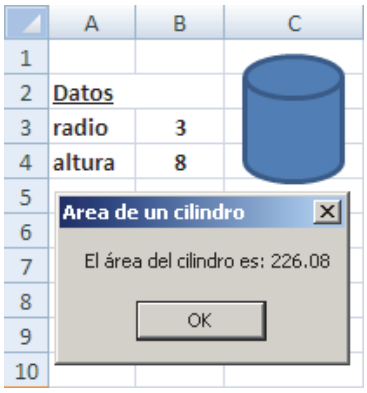
Ícono	Nombre
	Botón de comando
	Cuadro de lista
	Imagen
	Etiqueta
	Cuadro de texto

IV. Crea un programa en VBA que permita convertir grados Fahrenheit a Centígrados utilizando la siguiente fórmula: **C=(F-32)*5/9**.

V. Completa las siguientes expresiones:

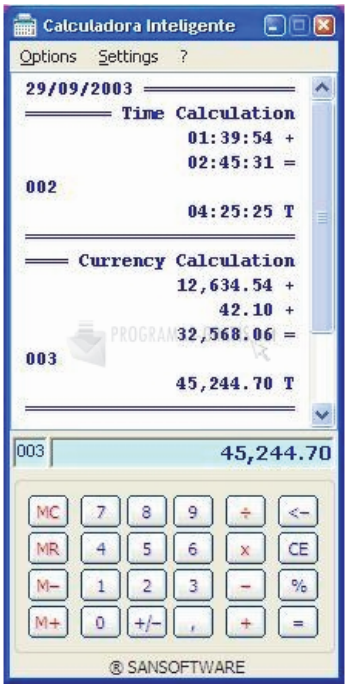
1. Un _____ está compuesto por controles.
2. Para cambiar el título de un formulario se debe utilizar la propiedad _____.
3. Un objeto es _____.
4. Excel graba las macros de forma predeterminada con un tipo de referencia_____.
5. Si deseas que una macro siempre esté disponible, debes guardarla en_____.

VI. Crea un programa en VBA que calcule el área de un cilindro, leyendo los datos de la hoja de cálculo tal como se muestra en la siguiente figura:



La calculadora con güincha

Descripción



Como parte del proyecto de este curso, desarrollarás una **calculadora con güincha**. Este tipo de calculadoras son frecuentemente utilizadas en las ventanillas bancarias, las cuales, aparte de permitir realizar operaciones matemáticas, pueden guardar dichas operaciones para que sean revisadas en cualquier momento por el cajero, sirviéndole como ayuda para realizar el cuadre de su caja.

Para este proyecto, tu calculadora debe tener las siguientes funcionalidades como mínimo:

1. Trabajar con las operaciones: adición, sustracción, multiplicación, división, porcentaje.
2. Debe permitir hacer dichas operaciones utilizando el *mouse* o el teclado.
3. Debe permitir guardar las operaciones que se van realizando.
4. Debe permitir limpiar tanto la pantalla de ingreso de números como de las operaciones guardadas.
5. Debe permitir exportar las operaciones a una hoja de Excel.

Cualquier funcionalidad adicional que puedas colocar a tu calculadora te permitirá obtener una mayor calificación en el proyecto.

Primera etapa

Para esta primera, debes hacer lo siguiente:

1. Crea tu archivo del proyecto.
2. Diseña el formulario colocando en él los controles que utilizarás para desarrollar la calculadora.

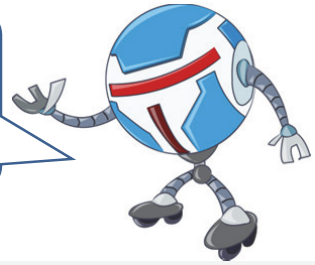
Anotaciones

- _____
- _____
- _____
- _____
- _____
- _____

CAPÍTULO 2

FUNDAMENTOS DE PROGRAMACIÓN

En este capítulo, conocerás nociones y conceptos básicos de programación en VBA, los cuales te permitirán desarrollar programas interesantes y de gran utilidad.



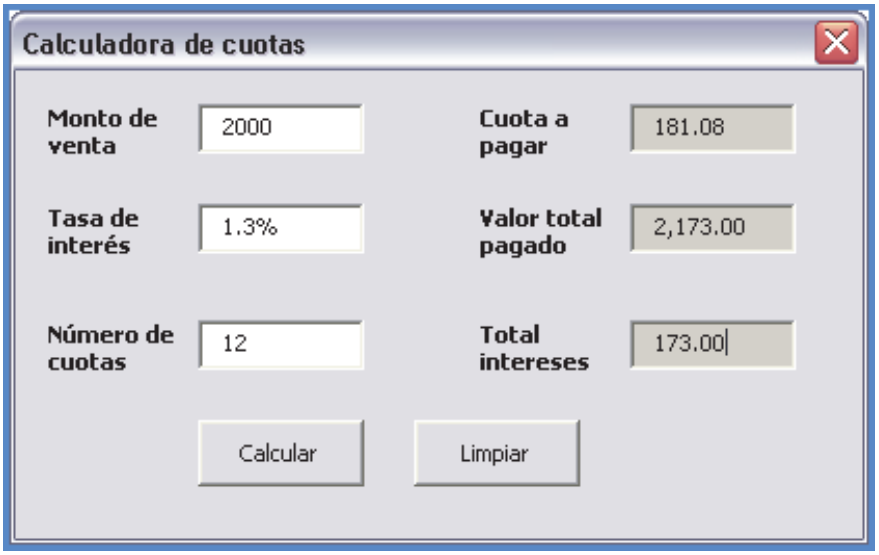
Tomado de: <http://commons.wikimedia.org/wiki/File:Centro-comercial.jpg>

José está de vacaciones en USA y ha realizado compras en diversos centros comerciales con su tarjeta de crédito Visa, por un monto de 2000 dólares. Sabiendo que su tasa de interés es del 1,3% y que él realizará el pago del total en 12 cuotas, necesita calcular su cuota mensual así como el total de intereses. ¿Es posible que lo haga utilizando Excel?

Tu Por supuesto, podemos obtener estos datos haciendo uso de las macros y del lenguaje de programación en VBA.

Con esta calculadora de datos, que elaborarás a lo largo de este capítulo, podrás resolver el problema de José y saber cuál es el monto a pagar en cada cuota, el total de intereses y el valor total pagado.

	A	B	C	D	E	F	G
1	Calculadora de cuotas						
2							
3	Monto de venta:	2000	soles		Cuota a pagar:	S/. 181.08	soles
4							
5	Tasa de interés:	1.3%			Valor total pagado:	S/. 2,173.00	soles
6							
7	Número de cuotas:	12			Total intereses:	S/. 173.00	soles
8							
9		Calcular		Limpiar			
10							



Antes de empezar con el contenido de este capítulo, te invitamos a que leas el siguiente artículo que te dará algunos alcances sobre lo que es un lenguaje de programación y reconocerás la importancia que tiene este para poder realizar el trabajo que te acabamos de presentar.

Lee atentamente:

Los ordenadores no hablan nuestro idioma, son máquinas y como tales, necesitan un lenguaje especifico pensado por el hombre para ellas. Además, necesitan constantemente interpretar todas las instrucciones que reciben. Dada la dificultad de comunicación insalvable entre el computador y el programador, pronto aparecieron lenguajes de programación que hacen posible la comunicación con el microprocesador, utilizando términos y símbolos relacionados con el tipo de problema que se debe resolver, mediante el empleo de herramientas que brinda la informática.

```
chaos% cat > wikipedia.pas
program wikipedia;
uses SysUtils;

begin
  WriteLn('Wikipedia is so cool');
end.
chaos% fpc wikipedia.pas
Free Pascal Compiler version 2.0.4 [2007/01/28] for i386
Copyright (c) 1993-2006 by Florian Klaempfl
Target OS: Linux for i386
Compiling wikipedia.pas
Linking wikipedia
6 Lines compiled, 0.2 sec
chaos% ./wikipedia
Wikipedia is so cool
chaos%
```

Estos lenguajes permiten, por un lado, escribir las operaciones que son necesarias realizar para resolver el problema de un modo parecido a como se escribiría convencionalmente (es decir, redactar adecuadamente el algoritmo de resolución del problema) y, por el otro, se encarga de traducir el algoritmo al lenguaje máquina (proceso conocido como compilación) con lo que se le confiere al programa la capacidad de correr (ser ejecutado) en el ordenador. El ordenador es en realidad tan sólo una máquina virtual, capaz de resolver todos los problemas que los usuarios seamos capaces de expresar mediante un algoritmo (programa).

En la actualidad hay muchos tipos de lenguajes de programación, cada uno de ellos con su propia gramática, su terminología especial y una sintaxis particular. Por ejemplo, existen algunos creados especialmente para aplicaciones científicas o matemáticas generales (BASIC, FORTRAN, PASCAL, etc.); otros, en cambio, se orientan al campo empresarial y al manejo de textos y ficheros, es decir, son en realidad fundamentalmente gestores de información (COBOL, PL/1, etc.), o muy relacionados con el lenguaje máquina del ordenador (como el C y el ASSEMBLER).

Los ordenadores se programaban en lenguaje máquina pero las dificultades que esto conllevaba, junto con la enorme facilidad de cometer errores, cuya localización era larga y compleja, hicieron concebir, en la década de los 40, la posibilidad de usar lenguajes simbólicos. Los primeros en aparecer fueron los ensambladores, fundamentalmente consistía en dar un nombre (mnemónico) a cada tipo de instrucción y cada dirección (etiqueta). Al principio se hacía el programa sobre papel y, después se traducía a mano con la ayuda de unas tablas, y se introducían en la máquina en forma numérica, pero pronto aparecieron programas que se ensamblaban automáticamente.

Finalmente, un lenguaje de programación es una notación para escribir programas, a través de los cuales podemos comunicarnos con el hardware y dar así las órdenes adecuadas para la realización de un determinado proceso. Un lenguaje está definido por una gramática o conjunto de reglas que se aplican a un alfabeto constituido por el conjunto de símbolos utilizados.

Tomado de: "Todo sobre programación". *Taringa*. Fecha de consulta: 28/10/2011 <http://www.taringa.net/posts/info/12056421/Historia-de-la-programacion-_resumen-algunos-lenguajes_ .html>

Tu trabajo en este capítulo consistirá en elaborar dos programas para el cálculo de cuotas por compras al crédito en una tienda comercial, uno de ellos será utilizando una hoja de cálculo y el otro directamente en un formulario de VBA.

Para empezar con el desarrollo del capítulo, abre una hoja de Excel 2010 y crea la siguiente hoja de cálculo. Guárdala con el nombre **cuotas.xlsx**.

	A	B	C	D	E	F	G
1	Calculadora de cuotas						
2							
3	Monto de venta:		soles		Cuota a pagar:		soles
4							
5	Tasa de interés:		%		Valor total pagado:		soles
6							
7	Número de cuotas:				Total intereses:		soles

Esta será la hoja de cálculo que utilizarás para tu trabajo, en la cual se debe calcular la cuota a pagar, el valor total pagado y el total de intereses, utilizando el monto de venta, la tasa de interés y el número de cuotas.

2.1 Conceptos iniciales de algoritmos

Cotidianamente las personas nos enfrentamos a problemas que resolvemos realizando una serie de pasos, procedimientos o acciones que nos permiten resolver dicho problema.

Por ejemplo:

Un alumno tiene que elaborar un trabajo escolar en Excel, para ello debe planificar la secuencia de pasos que debe seguir para lograrlo, ¿podrías ayudarlo completando los pasos que se muestran a continuación?

1) Encender la computadora
2) _____
3) _____
4) Ingresar los datos del trabajo en la hoja de cálculo
5) _____
6) _____
7) Imprimir el trabajo

¿Por qué crees que es importante que el alumno haya definido esta secuencia de pasos?

_____

_____

2.1.1 Concepto de algoritmo

Un algoritmo es un conjunto de pasos definidos y organizados que describe el proceso que se debe seguir, para dar solución a un problema determinado.

El algoritmo debe ser legible, correcto, modular, eficiente, ordenado, conciso y, de ser posible, que se desarrolle en el mínimo de tiempo.

2.1.2 Analizando el problema

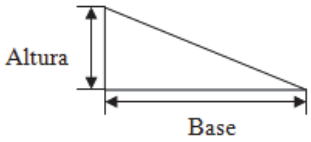
Es recomendable hacer un breve análisis del problema antes de aplicar algún algoritmo para poder resolverlo. Una forma simple de hacer esto es con una breve descripción utilizando el lenguaje natural.

Por ejemplo:

Calcular del área de un triángulo rectángulo en función de la base y altura.

1. **Descripción del problema**: se desea calcular el área de un triángulo.
2. **Datos que se necesitan para el cálculo (datos de entrada)**: base, altura
3. **Respuesta a obtener (datos de salida)**: área
4. **Fórmulas a aplicar:** para calcular el área de un triángulo se aplica la siguiente fórmula:

Área = base x altura



Hazlo tú mismo

Abre la hoja de cálculo **cuotas.xlsx** y completa el análisis del problema.

1) Escribe una breve descripción del problema:

_____

2) ¿Cuáles son los datos de entrada?

_____

3) ¿Cuáles son los datos de salida?

_____

4) Las fórmulas que debes usar serán las siguientes:

Sean: **m:** monto, **i:** porcentaje de interés, **n:** número de cuotas

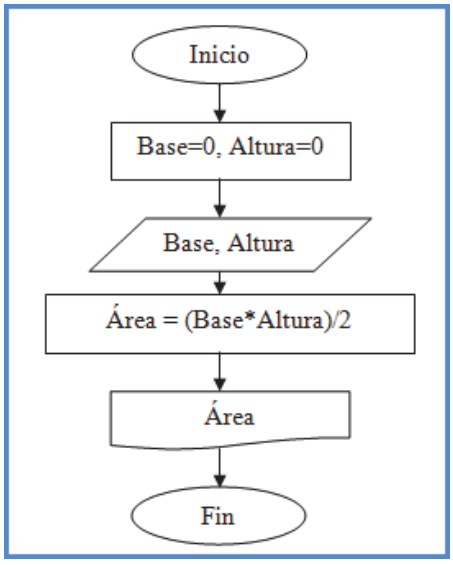
- **Cuota a pagar=**m x $(i(1+i)^n)/((1+i)^n-1)$
- **Valor total pagado**=n x cuota a pagar
- **Total intereses**=valor total pagado-m

Luego de analizar el problema, debes diseñar la solución, para ello se pueden utilizar técnicas o métodos estándar como por ejemplo el uso de diagramas de flujo y pseudocódigo.

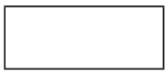
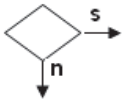
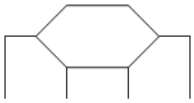
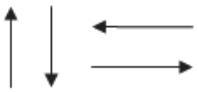
2.1.3 Diagramas de flujo

El diagrama de flujo o diagrama secuencial se utiliza para visualizar la secuencia detallada que comprende una tarea. Además, sigue una secuencia lógica de las actividades programadas.

Por ejemplo: diagrama de flujo para calcular el área de un triángulo



Símbolos utilizados en los diagramas de flujo

Representación del símbolo	Descripción
	Este símbolo marca el inicio y fin de un diagrama de flujo.
	Se utiliza para definir los datos de entrada.
	Representa un proceso.
	Representa una decisión (si algo se cumple, seguir la S; si no fuera el caso, seguir la N).
	Símbolo utilizado para representar una decisión múltiple. En su interior se almacena una expresión condicional y, dependiendo del valor de dicha expresión, se sigue por uno de los caminos alternativos.
	Este gráfico representa la impresión de un resultado.
	Las flechas indican la dirección del flujo del diagrama.

2.1.4 Pseudocódigo

Permite describir un algoritmo en un lenguaje simplificado sin depender de uno de programación. Es un lenguaje intermedio entre el humano y el de programación.

Por ejemplo: pseudocódigo para calcular el área de un triángulo

```
Inicio
    Inicializa base, altura
    Leer base, altura
    area=(base*altura)/2
    Imprimir "El área es : ", area
Fin
```

Cualquier método de diseño o representación de un algoritmo te permitirá, más adelante, escribir un programa de computadora en cualquier lenguaje de programación.

Ejercicio de aplicación 1

1. Realiza los diagramas de flujo para resolver los siguientes problemas:
   a) Sumar los “n” primeros números naturales
   b) Calcular el promedio de 4 números naturales
   c) Calcular las raíces de una ecuación de segundo grado

2. Realiza la representación en pseudocódigo de los problemas indicados en la pregunta anterior.

Hazlo tú mismo

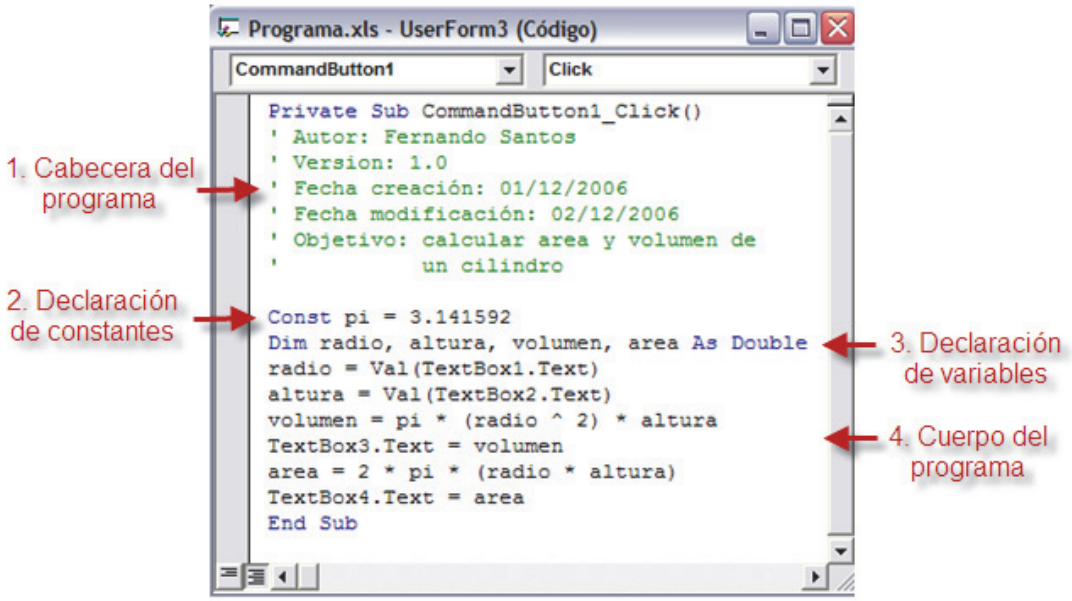
Siguiendo con el desarrollo de tu trabajo y en base a la información de la hoja de cálculo **cuotas.xlsx**, elabora el diagrama de flujo y pseudocódigo para el cálculo de cuotas de una venta al crédito.

Hasta este momento has elaborado el algoritmo y el diseño correspondiente al programa que vas a desarrollar, ahora irás construyendo tu programa.

2.2 El programa y sus partes

Un programa es un conjunto de instrucciones que se envían a una computadora para que realice una o varias acciones.

Se compone básicamente de las siguientes partes:



2.2.1 Cabecera del programa

Define el objetivo del programa comentándolo, es decir, se utiliza para documentar el programa, lo que las personas suelen dejar en segundo plano. Dentro del comentario, se incluye lo siguiente:

- Autor
- Versión actual
- Fecha del período de inicio o creación o de la última modificación del programa
- El objetivo del programa
- Otros detalles que ayuden a documentar el programa

Como habrás observado, para colocar un comentario en cualquier parte de tu programa, solo debes anteponer el carácter apóstrofe (') antes del comentario y presionar *Enter* para terminarlo, así por ejemplo:

```
'Este es un comentario
```

Declaración de constantes

Como su nombre lo dice, una constante es un valor que nunca cambia. Para declarar una constante se utiliza la variable **Const**, siguiendo la sintaxis detallada a continuación:

Const nombre_constante **[As** ***tipo_dato*****] = valor**

Ejemplo: Const Pi = 3.1415

Recuerda que la constante debe tener un valor asignado.

Declaración de variables

Todo programa consta de variables que facilitan la comprensión y orden del programa. Para poder declarar una variable, debes seguir la siguiente sintaxis:

Dim variable1 **As** ***tipo_dato1***, variable2 **As** ***tipo_dato2*****, ... ,** variableN **As** ***tipo_datoN***

Aquí algunos ejemplos de declaración de variables:

Dim edad as Integer

Dim nombre as String

Dim resultado as Boolean

Integer, String y Boolean son tipos de datos manejados por VBA. Más adelante, profundizarás tus conocimientos acerca del manejo de variables y tipos de datos.

Hazlo tú mismo

En base a lo mencionado anteriormente, ¿podrías explicar la diferencia entre una constante y una variable dentro de un programa?

_____
_____
_____
_____

Cuerpo del programa

Se le conoce como bloque del programa y puede ser simple o complejo dependiendo de la aplicación que estás desarrollando.

2.3 Tipos de datos

Al desarrollar un programa, muchas veces se manejan datos de diferentes tipos, tales como números, letras, fechas, u otros, los cuales dependen del valor que ellos representan.

Por ejemplo:

- Nombre de una persona
- Tipo de cambio del dólar
- ¿Estás afiliado al seguro?
- Porcentaje de IGV

Escribe 2 ejemplos más:

- _____
- _____

VBA maneja los datos agrupándolos de acuerdo a los siguientes tipos:

2.3.1 Datos numéricos

Están conformados por los caracteres numéricos del 0 al 9 y los caracteres especiales: +, -, ( ), /, e (formato científico).

Los principales tipos de datos numéricos son:

- **Byte:** generalmente es utilizado para representar **números naturales**. Ocupa 1 byte desde 0 hasta 255.

- **Integer:** generalmente es utilizado para representar **números enteros**. Ocupa 2 bytes y puede tomar un valor desde -32768 hasta 32767.
- **Long**: generalmente es utilizado para representar **números enteros largos**. Ocupa 4 bytes y puede tomar un valor desde −2,147,483,648 hasta 2,147,483,647.
- **Single**: generalmente es utilizado para representar **números reales**. Ocupa 4 bytes y puede tomar un valor desde -3,4028235E+38 hasta -1,401298E-45, para números negativos, y desde 1,401298E-45 hasta 3,4028235E+38, para números positivos.

Ejercicio de aplicación 2

Coloca los siguientes datos numéricos en la columna correspondiente de la tabla de acuerdo al tipo al que pertenece:

45 -254 254 0 8
123.21 256 34.23 73246 -63.75

Byte	Integer	Long	Single

-87 2.6 65 -1 394823

2.3.2 Datos alfanuméricos

Un dato alfanumérico es aquel que está formado por letras o por letras y números a la vez.

Existen dos tipos de datos alfanuméricos:

- Carácter simple (por ejemplo: “a”)
- Cadena de caracteres (por ejemplo: “marco”, “A001”)

En el caso de VBA resulta igual si se cuenta con un solo carácter (carácter simple) o de una cadena de caracteres. El espacio que ocupe esta dependerá del número de caracteres con que cuente.

Para VBA, el tipo de dato alfanumérico es el **String**.

2.3.3 Datos fecha/hora

La fecha se almacena como números de serie y la hora como fracciones de decimales. Esta es la razón por la cual una fecha y hora pueden ser utilizadas para realizar operaciones de cálculo.

Sin embargo, estos datos pueden ser presentados de acuerdo con el formato que sean requeridos:

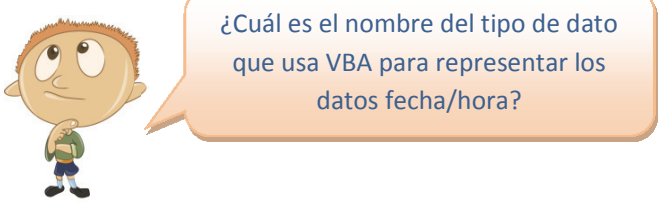
Fecha : DD/MM/AA o DD-MM-AA

Hora : HH:MM:SS AM/PM o HH:MM

Escribe dos ejemplos de datos fecha/hora con los formatos mostrados anteriormente:

- _____
- _____

Averigua…



Responde:

_____

2.3.4 Datos lógicos

Permite usar variables que tienen dos posibles valores: verdadero o falso. Por ejemplo:

- ¿Aprobó el examen? ¿Es mayor de edad?

Para VBA, el tipo de dato lógico es el **Boolean**.

Recuerda:

Dato	Tipo
Numérico	Byte, Integer, Long, Single
Alfanumérico	String
Fecha/hora	Date
Lógicos	Boolean

Variables en VBA

Cuando un programa solicita el ingreso de un dato, este será almacenado en un espacio de memoria de la computadora. A este espacio de memoria se le llama **variable.**

2.3.5 Declarando variables

Para poder utilizar variables dentro de un programa, estas deben ser declaradas al inicio del mismo con su respectivo nombre, tipo y siguiendo la sintaxis que viste anteriormente.

```
Dim variable1 As tipo_dato1, variable2 As tipo_dato2, ... , variableN As tipo_datoN
```

Reglas para nombrar una variable:

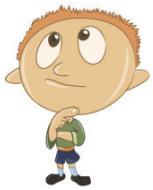
- Se pueden usar letras, números y algunos caracteres de puntuación, pero el primer carácter debe ser siempre alfabético.
- VBA no distingue entre mayúsculas o minúsculas, se recomienda seguir un estándar para hacerlo legible.
- No se pueden usar espacios ni puntos.
- No se pueden usar algunos caracteres especiales como # $% !
- Los nombres pueden contener hasta 254 caracteres de longitud.
- Existen palabras reservadas que no se pueden usar para nombres de variables o procedimientos.

Hazlo tú mismo

Escribe cómo declararías las siguientes variables en un programa:

Dato	Declaración de variable
monto de venta	
edad	
apellidos	
¿alcanzó vacante?	

Averigua…



Si no declaras el tipo de dato a una variable, ¿qué tipo de dato le asignará VBA?

Responde:

2.3.6 Asignando datos a las variables

Para asignarle un valor a una variable se usa el operador **"="**. Por ejemplo:

nombre="marco"

edad=15

Ejercicio de aplicación 3

Abre el archivo **cuotas.xlsx,** ingresa al editor de VBA y crea un módulo para la hoja de cálculo que estás utilizando.
Luego, crea el programa **Calcular** de la siguiente manera:

```
Sub Calcular()
…
End Sub
```

Finalmente, coloca dentro del programa lo siguiente:

1) La cabecera.
2) La declaración de variables y constantes que utilizarás.
3) Asigna a cada variable los datos leídos de las celdas de la hoja de cálculo.

Recuerda que, para leer un dato de una celda, se debe usar la función Range.
Por ejemplo: monto=Range("B3")

2.4 Tipos de operaciones en VBA

Muchas veces al elaborar un programa necesitarás realizar operaciones con los datos que posee.

Con VBA se pueden realizar los siguientes tipos de operaciones, las cuales dependen de los tipos de datos que tengan.

Operación	Tipo de dato
Aritmética	Byte, Integer, Long, Single
Lógica	Boolean
Concatenación	String

2.4.1 Operaciones aritméticas

Se utilizan para realizar operaciones aritméticas básicas como sumar, restar, multiplicar, dividir. Entre los operadores aritméticos tenemos:

Operador	Ejemplo	Tipo de operación	Resultado
+	5+2	Suma	7
-	5-2	Resta	3
-	-5	Negación (número negativo)	-5
*****	5*2	Multiplicación	10
/	5/2	División	2.5
^	5^2	Exponente (“elevado a la”)	25
Mod	18 mod 7	Obtiene el residuo de una división	4

Reglas para resolver operaciones aritméticas en VBA:

- Las operaciones son evaluadas de izquierda a derecha.
- El orden de precedencia de los operadores es el siguiente:

1. ()
2. ^
3. *, /
4. \
5. Mod
6. +, -

2.4.2 Operaciones lógicas

Se utilizan para establecer condiciones entre expresiones. Entre los operadores lógicos, tenemos:

Operador	Ejemplo	Tipo de operación	Resultado
And	(5<2) and (3>8)	Y (conjunción)	Falso
Or	(6>=4) or (2<1)	O (disyunción)	Verdadero
Not	Not(2>1)	NO (negación)	Verdadero

2.4.3 Operaciones de concatenación

Se utilizan para unir textos.

Operador	Ejemplo	Tipo de operación	Resultado
& o **+**	“milagros ” & “pérez”	concatenación	milagros pérez

Ejercicio de aplicación 4

Evalúa las siguientes operaciones y escribe el resultado correspondiente:

1) 18 mod 5 + 4*3 =
2) not(14/2-8>3+5*2)=
3) (6*3/2)^2=
4) (7*(10-5)mod 3)*4+9=

2.5 Desarrollando el programa

Ahora empieza a construir el cuerpo del programa. Para ello, debes ingresar las expresiones o fórmulas que te permitan calcular los valores necesarios y mostrarlos en las celdas correspondientes.

Hazlo tú mismo

Realiza las siguientes acciones en tu archivo **cuotas.xlsx**.

1) Ingresa al editor de VBA y elabora el cuerpo del programa **Calcular**, utilizando las fórmulas que viste en la etapa de análisis:

   Sean:

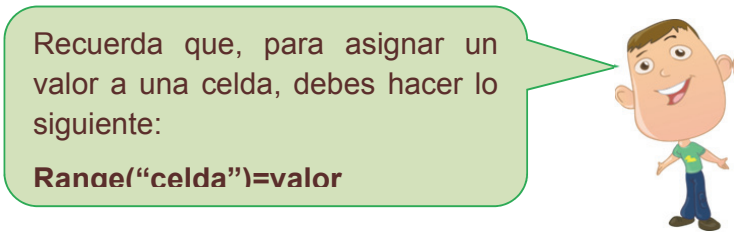
   m:monto, **i:**porcentaje de interés, **n:**número de cuotas

   Cuota a pagar=m x $(i(1+i)^n)/((1+i)^n-1)$

   Valor total pagado=n x cuota a pagar

   Total intereses=valor total pagado-m

2) Ingresa también el código que te permita colocar los resultados obtenidos en las celdas correspondientes.



Coloca el código completo de tu programa aquí:

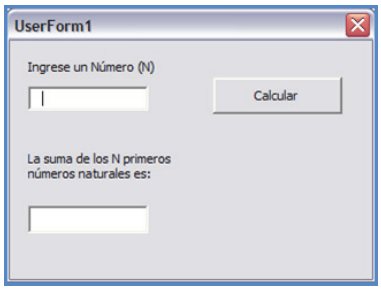
3) En la hoja de cálculo, crea un botón y asígnalo a tu programa **Calcular**.
4) Crea un programa adicional que te permita limpiar los valores de las celdas.

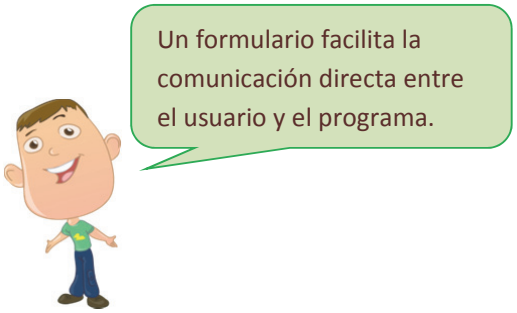
Tu trabajo debe quedar así:

	A	B	C	D	E	F	G
1	Calculadora de cuotas						
2							
3	Monto de venta:	2000	soles		Cuota a pagar:	S/. 181.08	soles
4							
5	Tasa de interés:	1.3%			Valor total pagado:	S/. 2,173.00	soles
6							
7	Número de cuotas:	12			Total intereses:	S/. 173.00	soles
8							
9		Calcular			Limpiar		
10							

2.6 Trabajando con formularios

Ahora vas a desarrollar un programa similar al del caso anterior, pero a través del uso de formularios.





Los lenguajes de programación ofrecen una extensiva variedad de controles que se pueden ir agregando de tal manera que el formulario utilizado sea lo más sencillo posible.

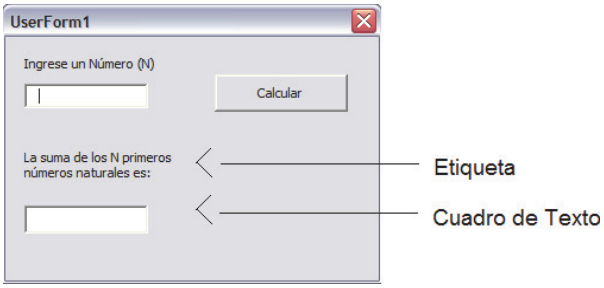
Antes de elaborar tu programa con formularios, debes conocer el uso de los principales elementos de la caja de controles que puedes usar en un formulario.

2.6.1 Etiquetas

Se utilizan para visualizar el texto que no se puede editar. De este modo, el usuario puede reconocer otros controles.

2.6.2 Cajas de texto

Permite que el usuario ingrese una información a la aplicación y que esta sea visualizada por él.



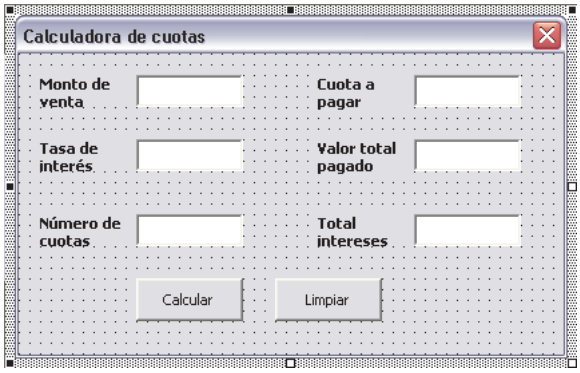
2.6.3 Botones de comando

Permiten ejecutar una orden. Los botones de comandos más utilizados son los siguientes:

Hazlo tú mismo

Ahora sí comenzarás a elaborar tu programa de Cálculo de cuotas, utilizando un formulario, para ello abre una hoja de Excel y realiza las siguientes acciones:

1) Ingresa al editor de VBA.
2) Inserta el siguiente formulario.

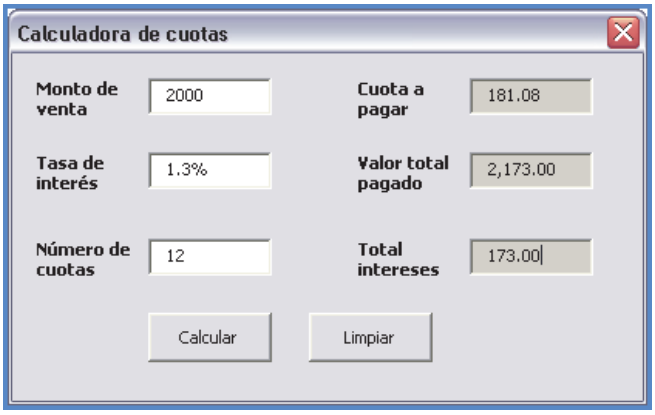


3) Haz doble clic en el botón **Calcular** e ingresa el código que utilizaste para el programa del cálculo de cuotas que realizaste anteriormente.

4) Modifica el código del programa teniendo en cuenta que tanto los datos de entrada como los datos de salida ya no se realizarán desde las celdas de Excel, sino desde las cajas de texto del formulario.

5) Agrega el código necesario al evento **OnClick** del botón **Limpiar** para que te permita borrar el contenido de las cajas de texto.

Al ejecutar tu programa, debe quedar así:



Ejercicio de aplicación 5

Crea los siguientes programas utilizando formularios en VBA:

1) Un programa que te permita convertir la velocidad de kilómetros por hora a metros por segundo.

2) Un programa que solicite los coeficientes (a, b, c) de una ecuación de segundo grado (ax^2+bx+c) y permita mostrar las raíces.
Recuerda que, para calcular las raíces de una ecuación de segundo grado, se debe usar la siguiente fórmula: $(-b\pm(b^2-4ac)^{1/2})/2a$

¿CUÁNTO APRENDÍ?

I. Relaciona el tipo de dato que utilizarías para usar las siguientes variables:

Variable	Tipo de dato
Domicilio	Long
año_nacimiento	String
Habitantes	Single
nro_pisos	Integer
area_circulo	Byte

II. Contesta las siguientes preguntas:

a) ¿Cuáles son las etapas que debes seguir para desarrollar un programa? Explica.

_____

_____

b) En VBA, ¿qué sucede si no se declara una variable antes de utilizarla?

_____

c) ¿Qué valor tiene la variable P después de realizar las siguientes operaciones?:

```
P=5
X=2+P
P=3
P=P+X
```

Resultado P=

d) ¿Qué valor tienen las variables X y W después de ejecutar las siguientes operaciones de asignación?

```
X=4
W=6
Y=X+W
X=W+Y
W=X+W
```

Resultado X=

Resultado W=

e) ¿En qué casos es conveniente utilizar constantes en un programa?

_____

III. Coloca V (verdadero) o F (falso) en las siguientes expresiones:

1) Al declarar una constante es obligatorio colocar el tipo de dato. ( )
2) En VBA, si se tiene una variable Nombre y otra NOMBRE son iguales. ( )
3) Para concatenar cadenas se debe utilizar el operador "+". ( )
4) El operador Mid permite obtener el residuo de una división. ( )
5) Si en un programa se desea utilizar una variable para almacenar el sexo de una persona (M o F), se debería utilizar el tipo de dato String. ( )

IV. Con una X, marca las alternativas en donde se hace una correcta declaración de variables y constantes y tipos de datos:

a) Dim edad as integer, cantidad as integer
b) Dim monto as string
c) Const interes as single
d) Const tipo_cambio=2.89

V. Calcula el resultado de las siguientes operaciones:

Expresión	Resultado
65\7	
65 mod 7	
(13+3)+8*3 mod 5 +4*7	
12^2/3 mod 9-1	

VI. Crea el diseño de los siguientes problemas utilizando diagrama de flujo o pseudocódigo:

a) Leer 3 números y mostrarlos en orden creciente.
b) Calcular el precio total de un artículo conociendo su precio de venta sin IGV y sabiendo que el IGV es del 18%.
c) Calcular la hipotenusa de un triángulo rectángulo teniendo como datos los catetos.

VII. Crea un programa para resolver el problema de la pregunta VI. b leyendo los datos y mostrando los resultados en la hoja de Excel.

VIII. Crea un programa para resolver el problema de la pregunta VI. c leyendo los datos y mostrando los resultados en un formulario.

Segunda etapa

Ahora aplicarás las herramientas de análisis y diseño de programas que has aprendido en este capítulo, para poder elaborar la calculadora con güincha; para ello, debes realizar lo siguiente:

1. Abre el archivo de tu proyecto.
2. Coloca un nombre a cada uno de los controles que has colocado en tu formulario para que los puedas reconocer en la etapa de programación.
3. Aplica las propiedades que creas convenientes a cada uno de los controles para que se vean presentables, por ejemplo, fondo, color de letra, borde, etcétera.
4. Elabora un análisis simple de cómo desarrollarías tu programa.

Hasta este momento ya tienes tu formulario diseñado y el análisis correspondiente al problema, en el siguiente capítulo aprenderás algunas herramientas adicionales que te permitirán empezar con la etapa de programación de tu calculadora con güincha.

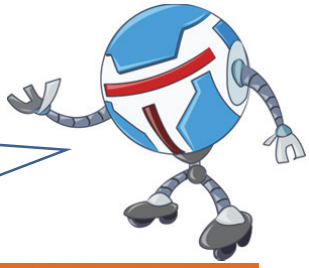
Anotaciones

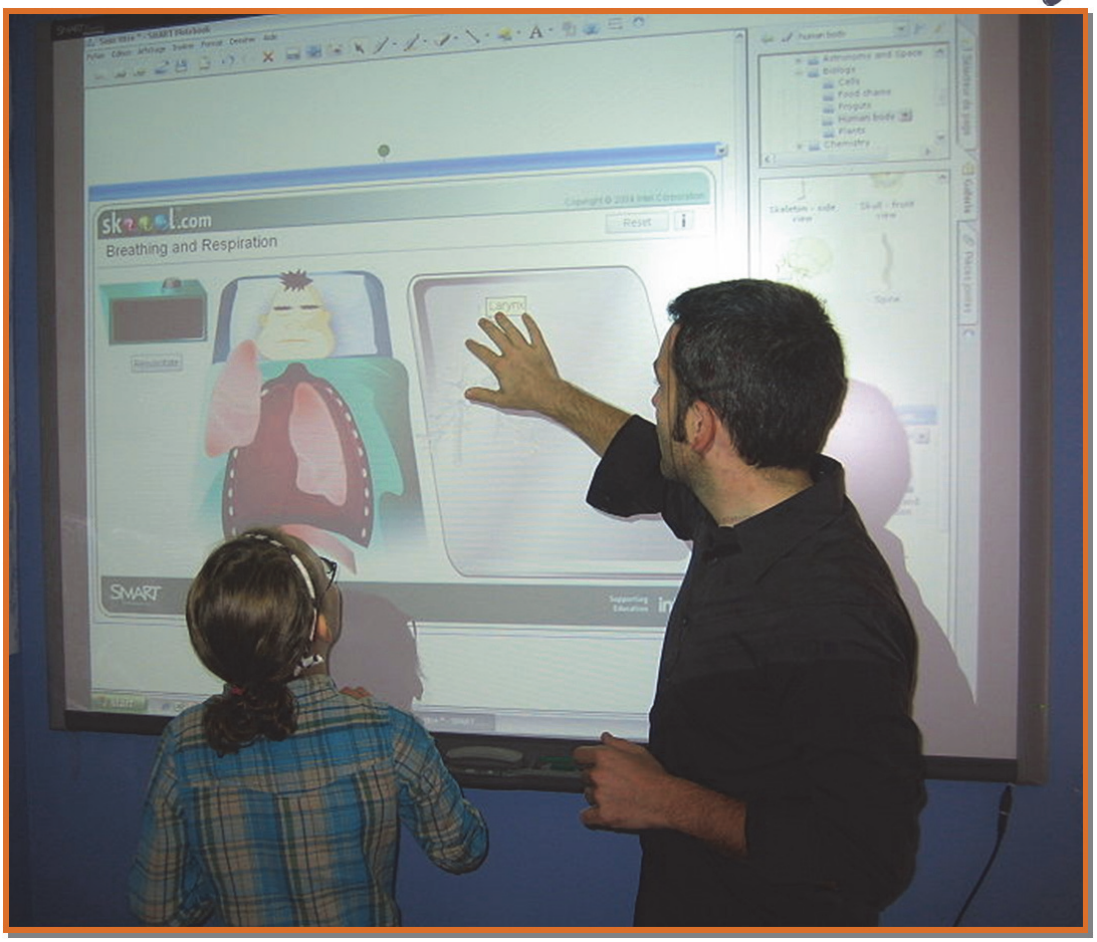
- _____
- _____
- _____
- _____
- _____
- _____

CAPÍTULO 3

PROGRAMACIÓN MODULAR

En este capítulo, aprenderás a desarrollar aplicaciones utilizando módulos y subprogramas en VBA, los cuales te permitirán dividir el desarrollo de tu aplicación en etapas que después se unirán para resolver el problema general.





Tomado de: Smartboard "Wikimedia commons". Fecha de consulta: 27/11/2011
http://commons.wikimedia.org/wiki/File:SmartBoard.JPG?uselang=es

Actualmente muchas escuelas están realizando una serie de innovaciones en la metodología de enseñanza, pues están aprovechando el desarrollo de las tecnologías de la informática y comunicación para el desarrollo de sus clases y el logro de las competencias deseadas en los alumnos. Estas constituyen una gran herramienta de apoyo.

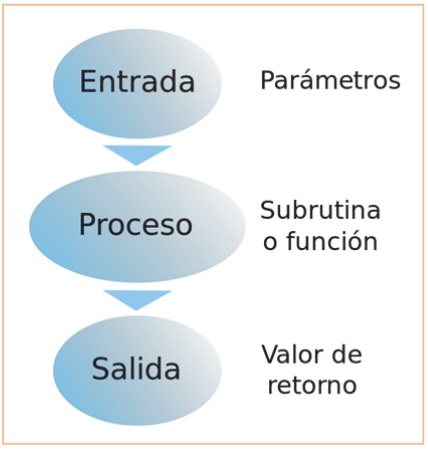
Supongamos que al profesor que aparece en la imagen se le ha pedido gestionar un programa con las notas de sus alumnos. ¿Qué es lo primero que debería hacer?

Primero debe determinar qué tareas va a realizar el programa, por ejemplo, permitir asignar la nota, modificarla, visualizarla según apellidos o de mayor a menor, etcétera. Una vez que ha esquematizado cada una de las tareas, empezará a trabajar en estas para finalmente obtener el producto deseado. A esto se le conoce como programación modular.

Antes de empezar con el contenido de este capítulo, te invitamos a que leas el siguiente artículo acerca de la programación modular.

La programación modular es un paradigma de programación que consiste en dividir un programa en módulos o subprogramas con el fin de hacerlo más legible y manejable.

Se presenta históricamente como una evolución de la programación estructurada para solucionar problemas de programación más grandes y complejos de lo que esta puede resolver.



Al aplicar la programación modular, un problema complejo debe ser dividido en varios subproblemas más simples, y estos a su vez en otros más simples. Esto debe hacerse hasta obtener subproblemas lo suficientemente simples como para poder ser resueltos fácilmente con algún lenguaje de programación.

Un **módulo** es cada una de las partes de un programa que resuelve uno de los subproblemas en que se divide el problema complejo original. Cada uno de estos módulos tiene una tarea bien definida y algunos necesitan de otros para poder operar. En caso de que un módulo necesite de otro, puede comunicarse con este mediante una interfaz de comunicación que también debe estar bien definida.

Tomado de: "Programación modular". *Wikimedia La enciclopedia libre*. Fecha de consulta: 20/11/2011 <http://es.wikipedia.org/wiki/Programaci%C3%B3n_modular>

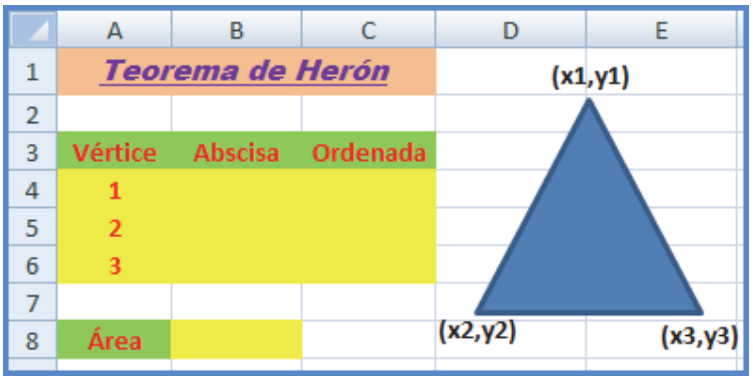
Tu trabajo

	A	B	C	D	E
1	Teorema de Herón			(x1,y1)	
2					
3	Vértice	Abscisa	Ordenada		
4	1	2	5		
5	2	7	3		
6	3	4	9		
7					
8	Área	12		(x2,y2)	(x3,y3)

```
Sub LeerVertices(x1, y1, x2, y2, x3, y3)
    x1 = Range("B4")
    y1 = Range("C4")
    x2 = Range("B5")
    y2 = Range("C5")
    x3 = Range("B6")
    y3 = Range("C6")
End Sub

Function CalcularLongitud(x1, y1, x2, y2) As Single
    CalcularLongitud = Sqr((x1 - x2) ^ 2 + (y1 - y2) ^ 2)
End Function

Function CalcularSemiPerimetro(lado1, lado2, lado3) As Single
    CalcularSemiPerimetro = (lado1 + lado2 + lado3) / 2
End Function

Function CalcularArea(p, a, b, c) As Single
    CalcularArea = Sqr(p * (p - a) * (p - b) * (p - c))
End Function

Sub Mostrar(area)
    Range("B8") = Round(area, 2)
End Sub

Sub Heron()
    Dim x1 As Single, x2 As Single, x3 As Single
    Dim y1 As Single, y2 As Single, y3 As Single
    Dim lado1 As Single, lado2 As Single, lado3 As Single
    Dim sp As Single, area As Single

    LeerVertices x1, y1, x2, y2, x3, y3
    lado1 = CalcularLongitud(x1, y1, x2, y2)
    lado2 = CalcularLongitud(x2, y2, x3, y3)
    lado3 = CalcularLongitud(x1, y1, x3, y3)

    sp = CalcularSemiPerimetro(lado1, lado2, lado3)
    area = CalcularArea(sp, lado1, lado2, lado3)
    Mostrar area

End Sub
```

En este capítulo, tu trabajo consistirá en elaborar un programa que te permita calcular el área de un triángulo conociendo sus coordenadas en el plano cartesiano y aplicando el teorema de Herón.

Para empezar con el desarrollo del capítulo, abre una hoja de Excel 2010 y crea la siguiente hoja de cálculo habilitado para macros. Guárdala con el nombre **heron.xlsm**.



Esta será la hoja de cálculo que utilizarás para tu trabajo, en la cual se debe calcular el área del triángulo tomando como datos de entrada las coordenadas de sus vértices.

A continuación se muestra la definición del problema a resolver:

El teorema de Herón permite calcular el área de un triángulo conociendo la longitud de sus lados utilizando la siguiente fórmula:

$$Área = \sqrt{p(p-a)(p-b)(p-c)}$$

Donde: a, b y c son los lados y p es el semiperímetro (la mitad del perímetro)

Como se mencionó anteriormente, en la hoja de cálculo solo se tienen como datos de entrada las coordenadas de los vértices del triángulo, por tanto, inicialmente debes calcular la longitud de cada uno de sus lados utilizando la siguiente fórmula:

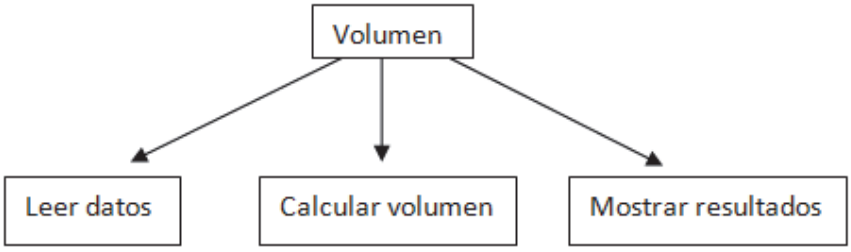
$$longitud = \sqrt{(x1-x2)^2 + (y1-y2)^2}$$

3.1 Diagrama de módulos

Un diagrama de módulos permite mostrar una vista gráfica de la estructura de módulos o partes en la que dividirás el desarrollo de tu programa. Es utilizado en la etapa de análisis.

Por ejemplo:

Un alumno ha elaborado el siguiente diagrama de módulos para utilizarlo en el desarrollo de un programa que le permita calcular el volumen de un cilindro.

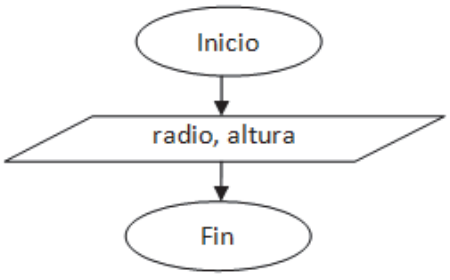


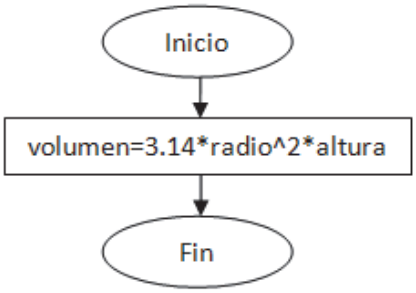
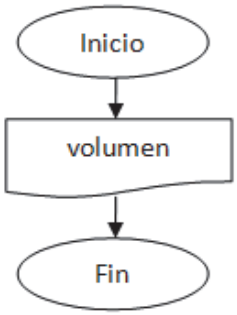
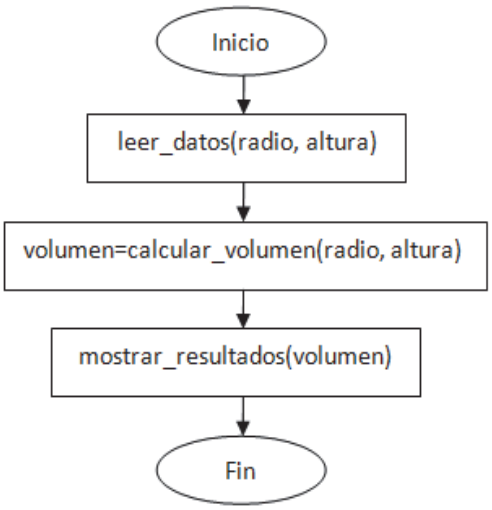
- ¿De cuántos módulos está compuesto el programa?_____
- ¿Cuáll es el nombre del módulo principal?_____
- ¿Por qué crees que es importante la programación modular?

  _____

Luego de definir los módulos que utilizarás en tu programa, deberás elaborar el diseño de cada uno de ellos, para lo cual utilizarás los métodos aprendidos en el capítulo anterior (diagrama de flujo o pseudocódigo).

Por ejemplo, a continuación se muestra el diagrama de flujo de los módulos definidos anteriormente para calcular el volumen de un cilindro.

 **leer_datos(radio, altura)**	Este módulo lee el radio y la altura del cilindro.

calcular_volumen(radio, altura) 	Este módulo calcula el volumen del cilindro utilizando como datos el radio y la altura.
mostrar_resultados(volumen) 	Este módulo muestra el resultado del volumen.
volumen() 	Este es el módulo principal, el cual usa los módulos anteriores. Aquí se invoca al módulo **leer_datos**. Aquí se invoca al módulo **calcular_volumen** usando el radio y la altura leídos. Aquí se muestra el resultado del volumen del cilindro.

Cuando diseñas en pseudocódigo o diagrama de flujo, los nombres de variables y módulos no deben tener **espacios en blanco**. También recuerda que, en diseño y programación, las palabras usadas no deben llevar **tilde**.

Ejercicio de aplicación 1

1) Una cajera de una tienda comercial necesita un programa que le permita calcular el subtotal, IGV y total de una venta. Para ello, cuenta con la hoja de Excel **venta.xlsx**, la cual se encuentra en la plataforma.

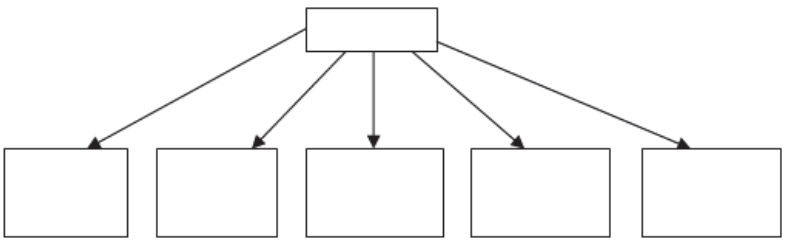
	A	B	C	D	E
1					
2	cantidad	precio	subtotal	IGV	total
3	15	S/. 4.00			

Se te pide hacer lo siguiente:

a) Elabora el diagrama de módulos que permita resolver el problema.
b) Elabora el diseño en pseudocódigo de cada uno de los módulos que has definido para resolver el problema mencionado.
c) Descarga de la plataforma el archivo **venta.docx** para revisar la solución a las dos preguntas anteriores.

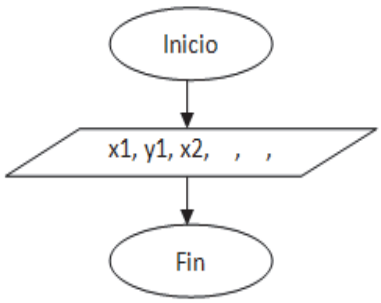
2) Completa el siguiente diagrama de módulos para poder resolver tu trabajo de este capítulo: "cálculo del área de un triángulo conociendo sus coordenadas en el plano cartesiano, utilizando el teorema de Herón".

Calcular semiperimetro	Mostrar area	Leer vertices
Calcular longitud	Calcular area	Heron

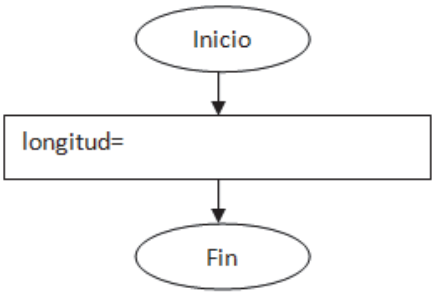


3) Completa el diagrama de flujo de los módulos que utilizarás para resolver tu trabajo.

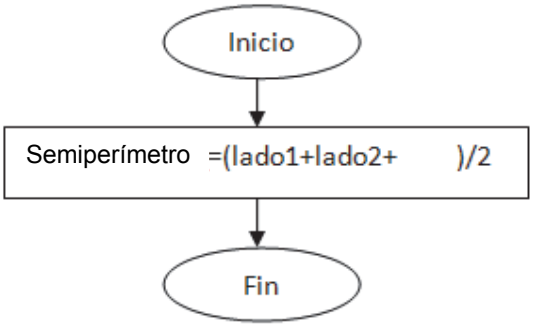
leer_vertices(x1,x2,y1,y2,x3,y3)



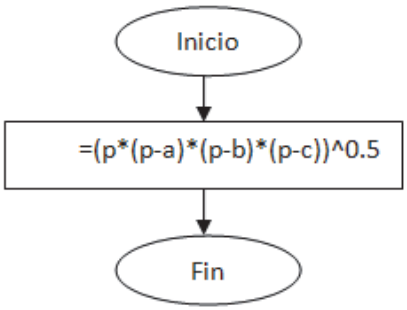
calcular_longitud(x1,y1,x2,y2)



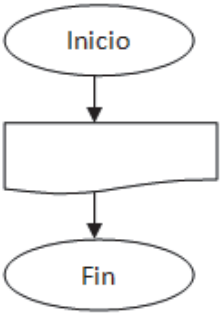
calcular_semiperimetro(lado1,lado2,lado3)

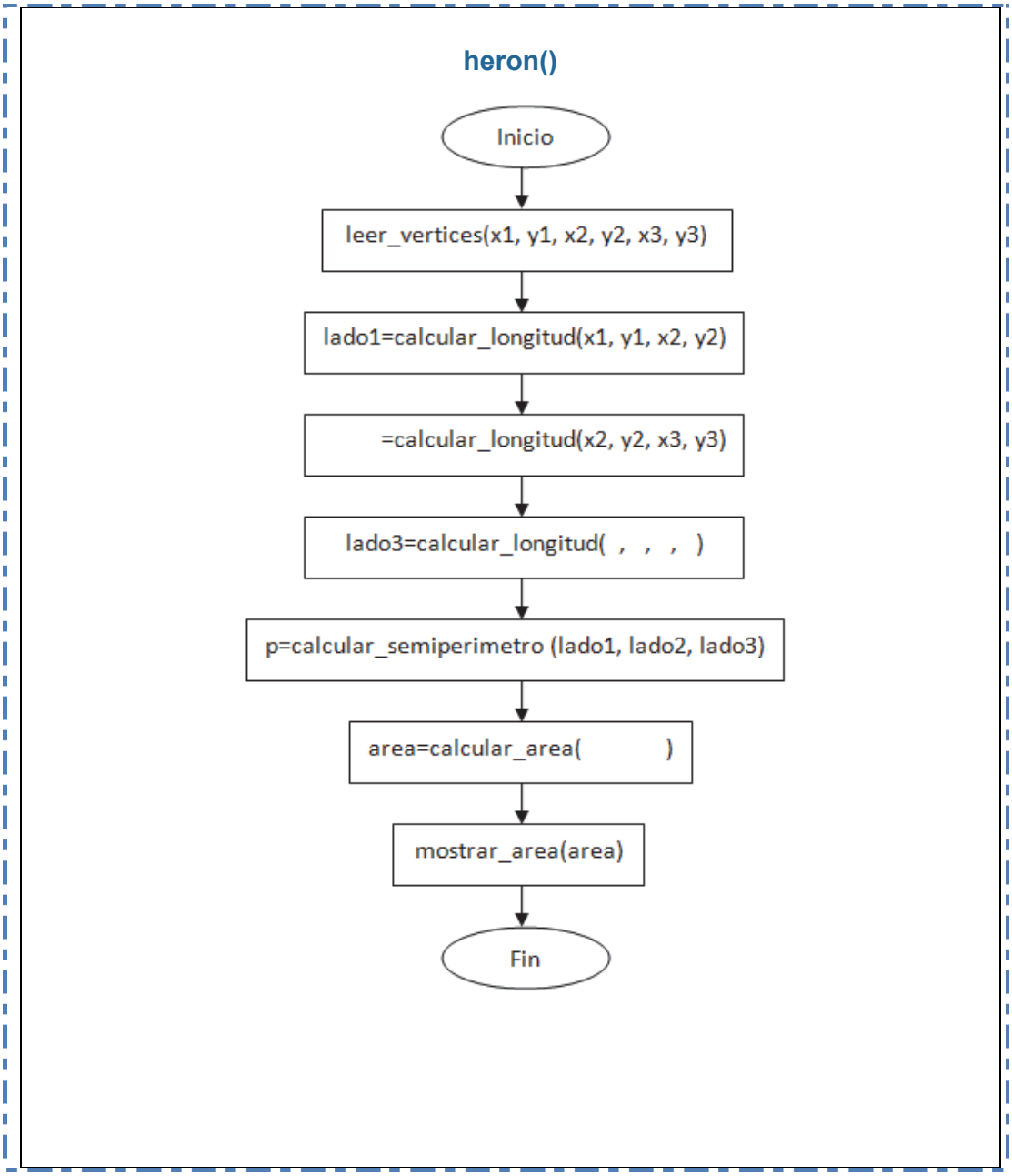


calcular_area(p,a,b,c)



mostrar_area(area)





3.2 Subprogramas en VBA

Como resultado de lo que viste en el apartado anterior, la programación modular se basa en la filosofía de “divide y vencerás”, es decir, se divide el problema planteado en problemas más simples, y cada uno de estos será implementado en módulos independientes, los cuales en el entorno de VBA se conocen como **subprogramas**.

Un subprograma es un conjunto de instrucciones o código fuente en VBA, que te permite realizar una tarea específica.

En VBA existen dos tipos de subprogramas:

- Función
- Procedimiento

3.2.1 Función

Es un subprograma diseñado para realizar una tarea específica. Una función toma una lista de valores llamados argumentos o parámetros y devuelve un único valor.

Al desarrollar un programa en VBA, puedes crear tus propias funciones o utilizar las funciones proporcionadas por el editor de VBA y Excel.

Esta es la sintaxis para el uso de funciones en VBA:

```
Function <nombre>( arg1, arg2, ... , argn) as tipo
    <sentencias> 'instrucciones que forman parte de la función
<nombre> = <resultado>  'Valor devuelto por la función
End function
```

Donde: arg1, arg2, arg3, …., argn son los parámetros o argumentos de la función.

Por ejemplo:
Función que calcula el promedio de 3 notas.

```
Function nota_promedio(nota1, nota2, nota3) As Single
    Dim p As Single

    p = (nota1 + nota2 + nota3) / 3
    nota_promedio = p

End Function
```

- ¿Cuál es el nombre de la función anterior?_____
- ¿Cuáles son sus parámetros?_____
- ¿Qué tipo de dato devuelve?_____

Las funciones no pueden ser ejecutadas directamente desde el editor de VBA, para ello es necesario definir un procedimiento que las invoque.

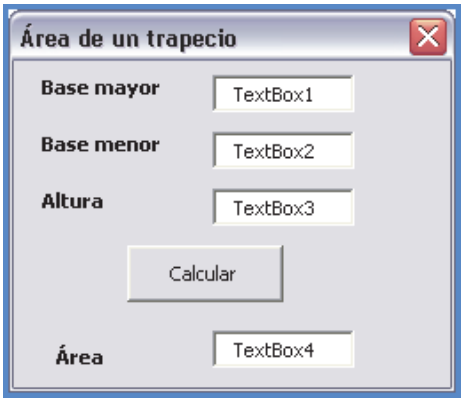
3.2.2 Invocando a una función

Invocar a una función se define como la llamada o uso que se hace a una función desde dentro de un subprograma. Para ello, debes digitar su nombre, sus argumentos entre paréntesis y, finalmente, asignar el resultado a una variable.

variable_resultado=**nombre_funcion**(arg1, arg2, arg3,….., argn)

Por ejemplo:

Programa que usa una función que calcula el área de un trapecio.



Recuerda que en este caso necesitas insertar un **formulario** en VBA.

```
Function trapecio(basemayor, basemenor, altura) As Double
  trapecio = ((basemayor + basemenor) * altura) / 2
End Function
Private Sub CommandButton1_Click()
    bmayor = Val(TextBox1.Text)
    bmenor = Val(TextBox2.Text)
    altura = Val(TextBox3.Text)
    TextBox4.Text = trapecio(bmayor, bmenor, altura)
End Sub
```

Se invoca a función **trapecio**.

Hazlo tú mismo

Abre el editor de VBA, elabora y prueba el programa mostrado anteriormente.

3.2.3 Procedimiento

Un procedimiento es la parte de un programa que realiza una o más tareas relacionadas y que tiene su propio nombre. El procedimiento puede tener cero, uno o más argumentos y puede devolver cero o más resultados.

Esta es la sintaxis para el uso de procedimientos en VBA:

```
Sub  <nombre>( arg1, arg2, ... , argn)
    <sentencias> 'instrucciones que forman parte del procedimiento
End Sub
```

Donde: arg1, arg2, arg3, …., argn son los parámetros o argumentos del procedimiento.

Por ejemplo:

Procedimiento que usa una hoja de Excel y lee los datos necesarios para calcular el área de un trapecio.

```
Sub LeerDatos(bmayor, bmenor, altura)
    bmayor = Range("B2")
    bmenor = Range("B3")
    altura = Range("B4")
End Sub
```

3.2.4 Invocando a un procedimiento

Invocar a un procedimiento se define como la llamada o uso que se hace a un procedimiento desde dentro de un subprograma. Esto lo puedes hacer de dos formas:

- 1ra forma: **NombreProcedimiento** arg1, arg2, arg3,….., argn
- 2da forma: **Call NombreProcedimiento** (arg1, arg2, arg3,….., argn)

Por ejemplo:

Programa que calcula el área de un trapecio leyendo y mostrando los resultados utilizando una hoja de Excel.

	A	B
1		
2	Base mayor	
3	Base menor	
4	Altura	
5		
6	Área	

Recuerda que en este caso necesitas insertar un **módulo** en VBA, para lo cual debes seleccionar la opción **Insertar** del menú **Ver**

```
Sub LeerDatos(bmayor, bmenor, altura)
    bmayor = Range("B2")
    bmenor = Range("B3")
    altura = Range("B4")
End Sub

Function Trapecio(bmayor, bmenor, altura) As Double
    Trapecio = ((bmayor + bmenor) * altura) / 2
End Function

Sub Mostrar(area)
    Range("B6") = area
End Sub

Sub Principal()
    Dim bmayor As Double, bmenor As Double
    Dim altura As Double, area As Double

    LeerDatos bmayor, bmenor, altura
    area = Trapecio(bmayor, bmenor, altura)
    Call Mostrar(area)

End Sub
```

Se invoca a procedimiento **LeerDatos**.

Se invoca a procedimiento **Mostrar**.

El programa anterior:

- ¿Cuántos procedimientos posee?_____
- ¿Cuántas funciones posee?_____
- ¿Cuántos subprogramas posee?_____

Hazlo tú mismo

Abre el editor de VBA, elabora y prueba el programa mostrado anteriormente.

Averigua…

Cuando creas una macro, ¿qué tipo de subprograma se genera?

Responde:

_____

Una línea de código en VBA puede ser tan larga como desees, pero si quieres una mejor visibilidad, puedes hacer un cambio de línea dejando un espacio en blanco y a continuación colocar el carácter "_". Luego, continúa el resto del código en la siguiente línea.

Ejercicio de aplicación 2

Abre tu archivo **venta.xlsx** que descargaste anteriormente y realiza las siguientes operaciones:

1) Ingresa al editor de VBA.
2) Inserta un módulo.
3) Crea un subprograma **calcula_subtotal**, que reciba como parámetros la cantidad y el precio, y que devuelva como resultado el valor del subtotal.
4) Crea un subprograma **calcula_igv,** que reciba como parámetro el subtotal y calcule el valor del IGV.
5) Crea un subprograma **calcula_total,** que reciba como parámetros el subtotal y el IGV, y calcule el valor total de la venta.
6) Crea un subprograma **principal** que lea los datos de las celdas, calcule el subtotal, IGV y total de la venta; para ello, se debe invocar a los subprogramas anteriores.
7) Crea un subprograma **limpiar**, el cual te permita limpiar los valores de las celdas A3 y B3.
8) En la hoja de Excel, inserta un botón y asócialo a tu subprograma principal.
9) En la hoja de Excel, inserta un botón y asócialo a tu subprograma **limpiar**.
10) Prueba la ejecución de los subprogramas anteriores.

3.3 Funciones predefinidas en VBA

Son aquellas funciones que vienen incorporadas en VBA y que puedes utilizarlas o invocarlas desde dentro de tus subprogramas cuando lo necesites.

Estas funciones se dividen en tres grandes grupos:

- Funciones de cadena de texto
- Funciones matemáticas
- Funciones de fecha/hora

3.3.1 Funciones de cadena de texto

Una cadena de texto es un conjunto de caracteres numéricos y/o alfanuméricos.

Por ejemplo:

- “hola mundo”
- “AIV-261”
- “12345”

Recuerda que el valor de una cadena de texto debe ir entre comillas (“ “).

Posición de un carácter en una cadena

Se denomina posición de un carácter en una cadena al lugar que ocupa comenzando a contar a partir de uno.
Por ejemplo:

Cadena	V	i	s	u	a	l		B	a	s	i	c
Posición	1	2	3	4	5	6	7	8	9	10	11	12

A continuación conocerás las funciones más importantes para el manejo de cadenas de texto:

Función	Descripción
STR (número)	Convierte un número a una cadena de caracteres. Ejemplo: STR(132454) Devuelve la cadena “132454”.
VAL (cadena numérica)	Obtiene el valor de una cadena de números. Ejemplo: VAL(“32345”) Devuelve el número 32345.
LEFT (cadena, n)	Muestra los caracteres en una cadena según la cantidad que se asigna a partir de la izquierda. Ejemplo: Si *cadena* = “Visual Basic”, entonces LEFT (cadena, 7) devuelve como resultado = “Visual ”.
RIGHT (cadena, n)	Devuelve los últimos “n” caracteres. Ejemplo: Si *cadena* = “Visual Basic”, entonces RIGHT (cadena, 5) devuelve como resultado = “Basic”.
MID (cadena, m, n)	Extrae “n” caracteres a partir de la posición m. Si no especifica la longitud de caracteres a extraer, retira todos los caracteres desde la posición especificada hasta el final. Ejemplo: Si *cadena* = “Visual Basic”, entonces MID (cadena, 8,5) devuelve como resultado = “Basic”.
LEN (cadena)	Muestra el tamaño de la cadena. Ejemplo: Si cadena = “Visual Basic”, entonces LEN(cadena) devuelve como resultado = 12.

Función	Descripción
TRIM (Cadena)	Omite los espacios al lado izquierdo o derecho de una cadena, pero no borra los espacios dentro de la misma. Ejemplo: Si cadena = [ ][ ]H o l a[ ][ ] entonces TRIM(cadena) devuelve como resultado = H o l a .

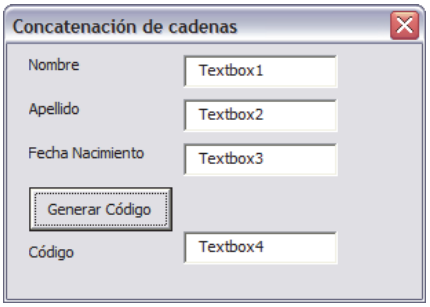
Hazlo tú mismo

Evalúa las siguientes operaciones con funciones de cadenas de texto y coloca los resultados.

Operación	Resultado
RIGHT(“Programación Modular”,7)	
MID(“Aprendiendo Excel Avanzado”,13,5)	
LEN(“InfoPUC”)	

Ejercicio de aplicación 3

Diseña el siguiente formulario:



Luego, coloca el código fuente necesario para que, al hacer clic en el botón **Generar Código,** se genere un código de usuario, tomando las dos primeras letras del nombre, las tres primeras letras del apellido así como el año de nacimiento del usuario.

3.3.2 Funciones matemáticas

Son funciones que operan sobre datos numéricos o devuelven datos numéricos. Entre las principales de ellas se encuentran:

Función	Descripción
Application.Pi()	Devuelve el valor de π (3,1416….)
SQR (Número)	Devuelve la raíz cuadrada de un número. Si variable = 9, entonces la función SQR (variable) devuelve como resultado = 3
ABS (Número)	Muestra el valor absoluto de un número. No tiene signo. Ejemplo: Si variable = -34.65, entonces la función **ABS** (variable) devuelve como resultado = 34.65 La función ABS es útil cuando se desea calcular la diferencia entre dos números, pero no se conoce cuál es el mayor. Ejemplo: Si X= 40; Y=60, entonces la función ABS(40-60) devuelve como resultado = 20
Rnd	Proporciona un número aleatorio en el rango 0<= x <1. Ejemplo: Rnd*10 +1 Devuelve números aleatorios entre 1 y 10.
INT (Número)	Devuelve el entero más grande de un número con decimales. Si deseas que devuelva con dos decimales, puedes utilizar la siguiente fórmula: X= INT(100*X)/100 Ejemplo: Si variable = 34.12, entonces la función INT (variable) devuelve como resultado = 34
Round(Número, decimales)	Redondea un número a un número de decimales especificado. Ejemplo: Round (14,4583,2) devuelve como resultado =14,46

Por ejemplo, el siguiente programa calcula el área de un círculo.

```
Sub Area_circulo()
    r = InputBox("Ingrese el radio")
    p = Application.Pi()
    a = p * r ^ 2
    a = Round(a, 2)

    MsgBox "El área del circulo es: " & a, vbDefaultButton1, "Area"

End Sub
```

Hazlo tú mismo

Abre el editor de VBA, elabora y prueba el programa mostrado anteriormente.

Ejercicio de aplicación 4

Diseña el siguiente formulario:

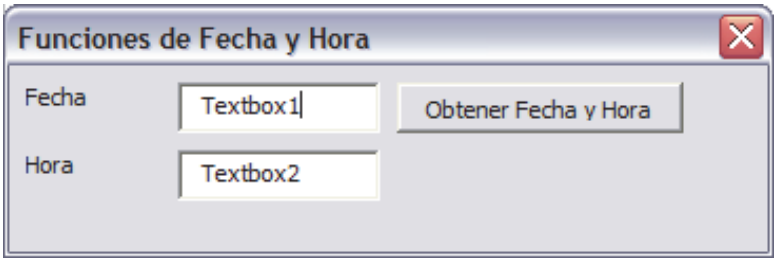
Luego, coloca el código fuente necesario para que al hacer clic en el botón **Generar,** se genere un número entero aleatorio entre 1 y 10.

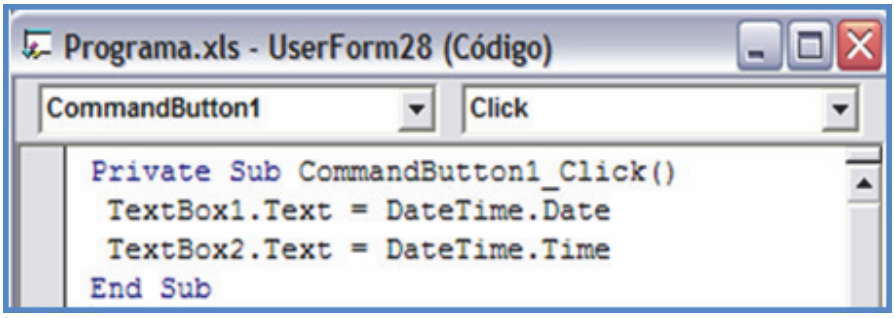
3.3.3. Funciones de fecha y hora

VBA cuenta con algunas funciones que permiten realizar cálculos con fechas y horas, entre las principales de estas funciones se tienen:

Función	Descripción
Date	Muestra la fecha actual, con formato MM/DD/YYYY Donde: MM – Mes; DD – Día; YYYY – Año
Time	Devuelve la hora actual en el formato HH:MM:SS.SSS Donde: HH- Horas; MM – Mes; SS – Segundos SSS – Centésimas de segundos

Por ejemplo, el siguiente programa devuelve la fecha y hora del sistema.





Hazlo tú mismo

Abre el editor de VBA, elabora y prueba el programa mostrado anteriormente.

Ejercicio de aplicación 5

Elabora el programa solicitado en tu trabajo de este capítulo. Para ello, debes seguir la siguiente secuencia:

1) Abre tu archivo **heron.xlsm.**
2) Ingresa al editor de VBA.
3) Inserta un módulo.
4) Crea los siguientes subprogramas basándote en el diagrama de flujo cuyo diseño se completó en el ejercicio de aplicación 1.
    a) Subprograma leer_vertices
    b) Subprograma calcular_longitud
    c) Subprograma calcular_semiperimetro
    d) Subprograma calcular_area
    e) Subprograma mostrar_area
    f) Subprograma heron
5) En la hoja de cálculo, inserta un botón y asócialo al subprograma principal.
6) Prueba la ejecución de tu programa

¿CUÁNTO APRENDÍ?

I. Contesta las siguientes preguntas:

a) ¿Cuál es la diferencia entre un procedimiento y una función?

_____

_____

b) ¿A qué se conoce como cadena de texto?

_____

c) ¿Para qué se usan los diagramas de módulos?

_____

d) Analiza las siguientes instrucciones:

```
Dim texto as String, cad as String
texto=”cadena de caracteres”
cad=MID(texto,11,LEN(texto)-10)
```

¿Cuál es el valor final de la variable **“cad”**?_____

II. Coloca V (verdadero) o F (falso) en las siguientes expresiones:

1) La función INT sirve para redondear un número a cero decimales. ( )
2) La función TRIM sirve para quitar los espacios en blanco a una cadena. ( )
3) Para invocar a una función se usa la palabra reservada **Call**. ( )
4) **Calcular valor** es un nombre correcto para un subprograma. ( )
5) Una función puede ser ejecutada directamente desde el editor de VBA. ( )

III. Elabora un programa que calcule el volumen de un cono, leyendo los datos de una hoja de Excel, tal como se muestra en la figura.

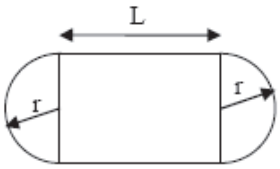
	A	B
1	**Volumen de un cono**	
2		
3	radio	3
4	altura	7
5		
6	volumen	65.9736

Recuerda que el volumen de un cilindro se calcula con la siguiente fórmula:

$$volumen = \frac{\prod * radio^2 * altura}{3}$$

IV. Un agricultor desea cercar su granja con alambre y, además, cubrir con abono el terreno que la comprende. Para calcular el costo de materiales, debe tomar en cuenta lo siguiente:

- El terreno tiene la forma de un rectángulo con 2 semicírculos a los extremos, tal como se indica en la figura:



- El alambre se vende por rollos completos de **P** metros, siendo **α** soles el costo del rollo.
- La cerca consta de 6 vueltas completas de alambre, en las cuales se usarán una cantidad **Q** de rollos. No debe sobrar una longitud de alambre mayor a **P** metros (tamaño de un rollo)
- Se estima que el costo del abono es de **β** soles por metro cuadrado de terreno.

Se necesita realizar un programa en VBA que permita calcular el costo del proyecto, para ello debes realizar lo siguiente:

1) Elabora el análisis del problema
2) Elabora el diagrama de módulos considerando los siguientes módulos:
   - CalcularPerimetro
   - CalcularArea
   - CalcularNumeroRollos
   - Principal
3) Elabora el diseño de cada uno de los módulos en diagrama de flujo.
4) Implementa el subprograma **CalcularPerimetro** que reciba como parámetros el radio de los semicírculos y el lado del rectángulo que forma la granja, y calcule el perímetro de la granja en metros.
5) Implementa el subprograma **CalcularArea** que reciba como parámetros el radio de los semicírculos y el lado del rectángulo que forma la granja, y calcule el área de la granja en metros cuadrados.
6) Implementa el subprograma **CalcularNumeroRollos** que reciba como parámetros el perímetro de la granja y la longitud del alambre que contiene cada rollo, y calcule el número de rollos de alambre completos necesario para cercar la granja.
7) Implementa el subprograma **Principal**, que lea los datos necesarios, use los subprogramas anteriores, calcule y muestre el costo total de materiales, tal como se muestra en la siguiente figura:

	A	B
1	Datos de Entrada	
2	Radio, r :	5
3	Lado, L:	20
4	Longitud de rollo, A:	30
5	α (soles/rollo):	100
6	β (soles/m2):	80
7	Costo (soles):	23783.2

Hoja1 Hoja2

V. Un alumno desea calcular el volumen de un lápiz, para lo cual divide su problema en tres secciones: la punta del lápiz (forma de cono), el cuerpo del lápiz (forma de cilindro) y el borrador del lápiz (forma de media esfera), tal como se muestra en la siguiente figura:

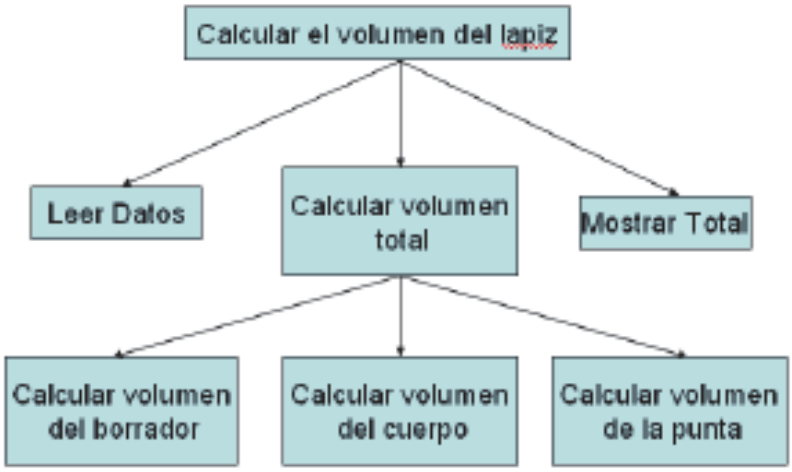
Las dimensiones del lápiz serán leídas desde una hoja Excel, similar a la siguiente:

	A	B
1	**Volumen del lápiz**	
2		
3	radio	4
4	altura punta	3
5	altura cuerpo	8
6		
7	volumen	

Se te pide realizar lo siguiente:

a) Elaborar el análisis del problema

b) Elaborar el diseño en pseudocódigo basándote en el siguiente diagrama de módulos



c) Elaborar la implementación en VBA de cada uno de los módulos

Tercera etapa

Ahora aplicarás el concepto de programación modular que has aprendido en este capítulo, para poder elaborar la calculadora con güincha; para ello, debes realizar lo siguiente:

1. Abre el archivo de tu proyecto.
2. Implementa los subprogramas necesarios para el funcionamiento de las operaciones de tu calculadora tales como adición, sustracción, multiplicación, etcétera.

Hasta este momento tu calculadora debería realizar las principales operaciones, en el siguiente capítulo aprenderás algunas herramientas adicionales que te permitirán concluir con la elaboración de tu calculadora con güincha.

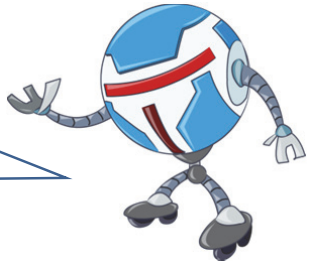
Anotaciones

- _____
- _____
- _____
- _____
- _____
- _____

CAPÍTULO 4

ESTRUCTURAS DE CONTROL

En este capítulo, aprenderás a incorporar estructuras de control a tus programas, las cuales te permitirán controlar la ejecución del flujo de instrucciones de los mismos.



Tomado de: "Decisions". Wikimedia Commons. Fecha de consulta: 28/11/2011
http://commons.wikimedia.org/wiki/File:Decisions,_decisions_-_geograph.org.uk_-_912859.jpg

En muchas oportunidades nos encontramos ante problemas o situaciones particulares, en las que tenemos que decidir las acciones a realizar de acuerdo a ciertas condiciones que nos permitan controlar la situación. De igual modo, al desarrollar un programa existen determinados pasos que se deben ejecutar dependiendo de condiciones del mismo programa. En este capítulo, aprenderás acerca de las estructuras de control.

Un robot japonés que aprende a tomar decisiones

Los investigadores del Grupo de Hasegawa en el Instituto de Tecnología de Tokio han creado un robot que es capaz de aplicar los conceptos aprendidos para realizar nuevas tareas, entender su entorno y para llevar a cabo las instrucciones que antes no sabía cómo hacer.

El robot tiene una tecnología neural detrás que le permite averiguar qué hacer en una situación dada mediante el almacenamiento de información en una red construida para imitar el cerebro humano.

Por ejemplo, el equipo demuestra su tecnología pidiendo al robot llenar un vaso con agua de una botella, lo que lo hace muy rápido y hábilmente. Esta parte no es nada nuevo, él l está simplemente siguiendo las instrucciones predefinidas. En la siguiente ronda, sin embargo, al robot se le pide que enfríe la bebida. Esta vez, el robot tiene que hacer una pausa para considerar lo que debe hacer para llevar a cabo esta nueva solicitud.

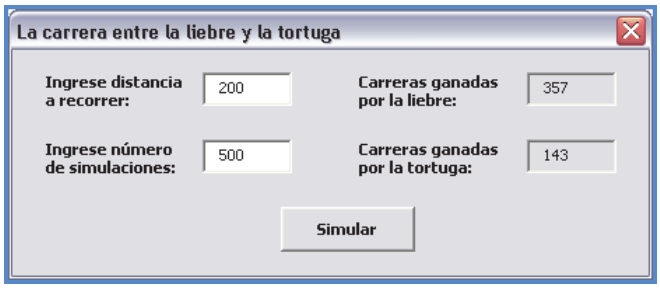
Se puede ver inmediatamente que no se puede llevar a cabo la solicitud de nuevo bajo las circunstancias actuales, ya que ambas de sus manos ya están siendo utilizadas (uno para sostener la copa y la otra la botella), de modo que coloca la botella hacia abajo y, luego, llega a recuperar un cubo de hielo y rápidamente lo deposita en la taza.

Esta pequeña demostración, si no es tan emocionante para ver, representa un verdadero salto adelante en la tecnología robótica y programación. Ser capaz de aprender significa que el robot puede ser programado con solo un conjunto muy básico de conocimientos previos que luego se construye sobre el tiempo en que el robot existe, sin necesidad de programación adicional, no muy diferente de cómo los seres humanos comienzan con muy poca información al nacer y van aprendiendo sobre lo que van experimentando y son capaces de hacer esto durante toda la vida.

Tomado de:Tecnoark Internet Noticias de ciencia tecnología actual. Fecha de consulta: 27/11/2011 http://tecnoark.com/un-robot-japones-que-aprende-a-tomar-decisiones/7101/.

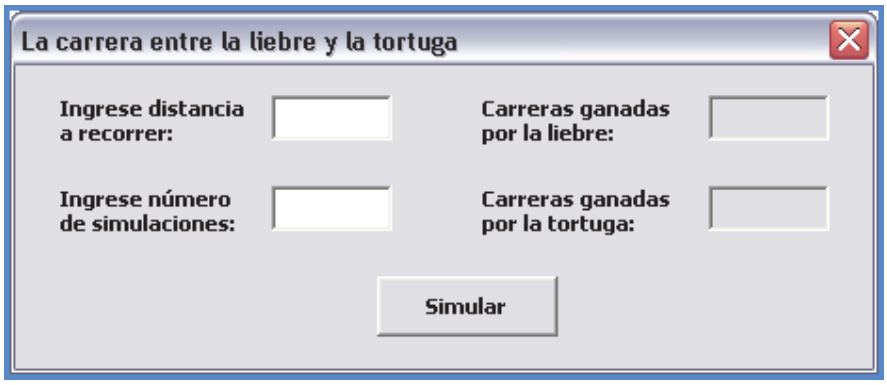
En este capítulo, tu trabajo consistirá en elaborar un programa que te permita simular la famosa carrera entre la liebre y la tortuga.

Tu trabajo




```
Sub principal()
    Dim numSimulaciones  As Integer
    Dim simul As Integer
    Dim liebre As Integer
    Dim tortuga As Integer
    Dim meta As Integer
    meta = Range("b1")
    numSimulaciones = Range("b2")
    liebre = 0
    tortuga = 0
    For simul = 1 To numSimulaciones
        If simulaCarrera(meta) = "liebre" Then
            liebre = liebre + 1
        Else
            tortuga = tortuga + 1
        End If
    Next
    Range("b3") = liebre
    Range("b4") = tortuga
End Sub


Function simulaCarrera(ByVal meta As Integer) As String
    Dim l As Integer
    Dim t As Integer
    l = 0
    t = 0
    Do While (l < meta) And (t < meta)
        l = l + avance("liebre", Rnd())
        t = t + avance("tortu", Rnd())
    Loop
    If (l > t) Then
        simulaCarrera = "liebre"
    Else
        simulaCarrera = "tortuga"
    End If
End Function



Function avance(ByVal p As String, ByVal v As Single) As Integer
    If (p = "tortuga") Then
        If (v <= 0.3) Then
            avance = 1
        Else
            avance = 2
        End If
    Else
        If (v <= 0.75) Then
            avance = 1
        ElseIf (v <= 0.9) Then
            avance = 2
        Else
            avance = 3
        End If
    End If
End Function
```

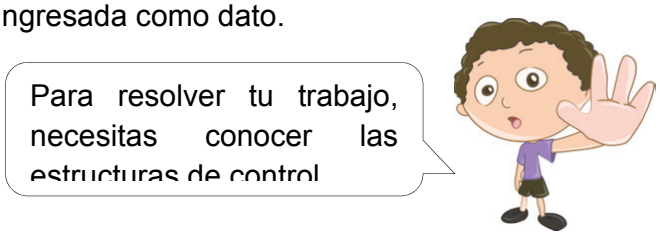
Para empezar con el desarrollo del capítulo, abre una hoja de Excel, ingresa al editor de VBA y diseña el siguiente formulario. Guarda tu archivo con el nombre **simulación.xlsm**.



Este será el formulario que utilizarás para tu trabajo, en el cual se debe calcular el número de carreras ganadas por la liebre y la tortuga de un total de carreras simuladas.

Para resolver el problema, debes tener en cuenta las siguientes condiciones:

- Se generarán avances por unidad de tiempo de diferente magnitud para la liebre y la tortuga durante toda la carrera.
- Ambos corredores partirán desde una misma posición inicial.
- La carrera terminará cuando uno de los participantes alcance o exceda la distancia ingresada como dato.



En el mundo de la programación, existen dos tipos de estructuras de control:

- Estructuras de control condicional
- Estructuras de control de ciclos repetitivos

4.1 Estructuras de control condicional

Son aquellas que permiten verificar si el programa cumple una determinada condición para que se puedan ejecutar un conjunto de instrucciones definidas.

Por ejemplo: si se desea imprimir el resultado final que obtuvo un alumno en el curso de computación, se debería llegar a la siguiente condición:

Si el promedio es mayor que 10.5 , entonces

Imprimir “Aprobado”

Caso contrario

Imprimir “Desaprobado”

En VBA, los principales tipos de estructuras condicionales son las siguientes:

- Condicional simple de la forma: **If….then**
- Condicional con contingencia de la forma: **If….then….else**
- Condicionales anidadas
- Condicional múltiple de la forma: **Select….case**

A continuación verás cada uno de los tipos de estructuras condicionales con mayor detalle.

4.1.1 Condicional simple de la forma: *If…then*

Se utiliza cuando se necesita realizar un conjunto de acciones establecidas siempre y cuando se cumpla una condición.

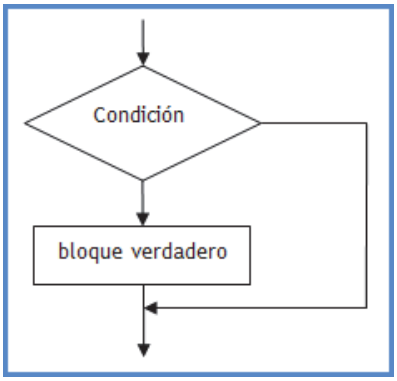
Su sintaxis es la siguiente:

```
If <expresión booleana> then
        <sentencias>
End If
```

El resultado de una expresión booleana puede ser **TRUE** (verdadero) o **FALSE** (falso)

Lo cual quiere decir que si se cumple la expresión booleana, se realizan todas las sentencias escritas después del **then**.

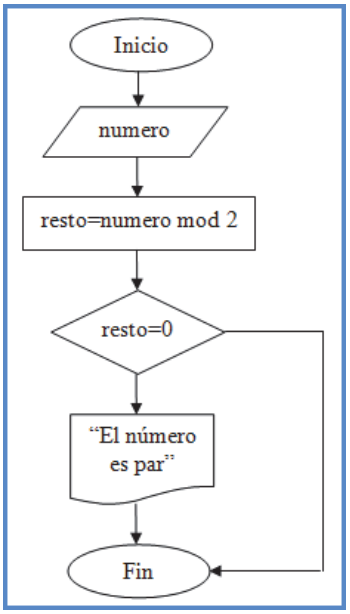
Su representación en diagrama de flujo es la siguiente:



Observa el siguiente ejemplo de aplicación:

Se desea diseñar e implementar un programa que permita verificar si un número es par o impar.

Diagrama de flujo



Diseño del formulario

El programa solicitará el ingreso del número y al hacer clic en el botón **Verificar**, se mostrará un MessageBox indicando si el número es par.

Implementación del programa

```
Private Sub CommandButton1_Click()
    Dim resto As Integer

    resto = Val(TextBox1.Text) Mod 2

    If (resto = 0) Then
        MsgBox "El número es par", vbDefaultButton1, "Resultado"
    End If

End Sub
```

Hazlo tú mismo

Elabora y prueba el programa mostrado anteriormente utilizando VBA.

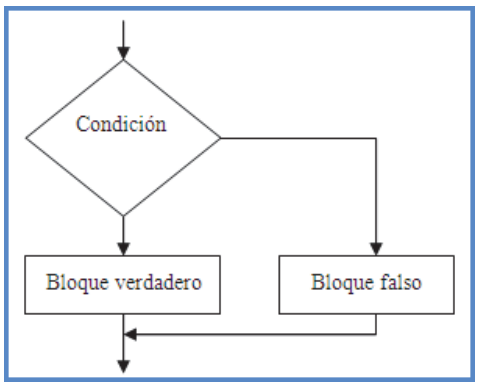
4.1.2 Condicional con contingencia de la forma: *If…then…else*

Se utiliza cuando se ha planificado la ejecución de un grupo de instrucciones cuando la condición es verdadera, y otro grupo de instrucciones cuando la condición es falsa.

Su sintaxis es la siguiente:

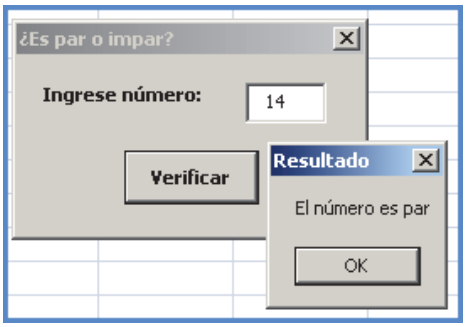
```
If <expresión booleana> then
        <sentencias 1>
Else
        <sentencias 2>
End If
```

Su representación en diagrama de flujo es la siguiente:



Hazlo tú mismo

Modifica el diagrama de flujo, diseño del formulario e implementación del programa anterior para que muestre si el número ingresado es par o impar.



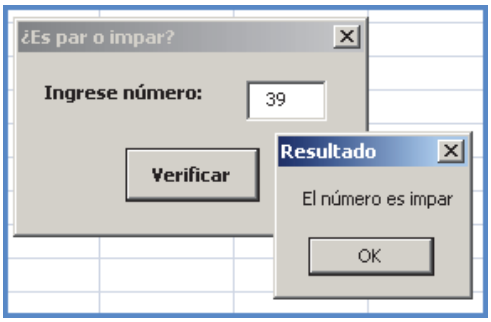


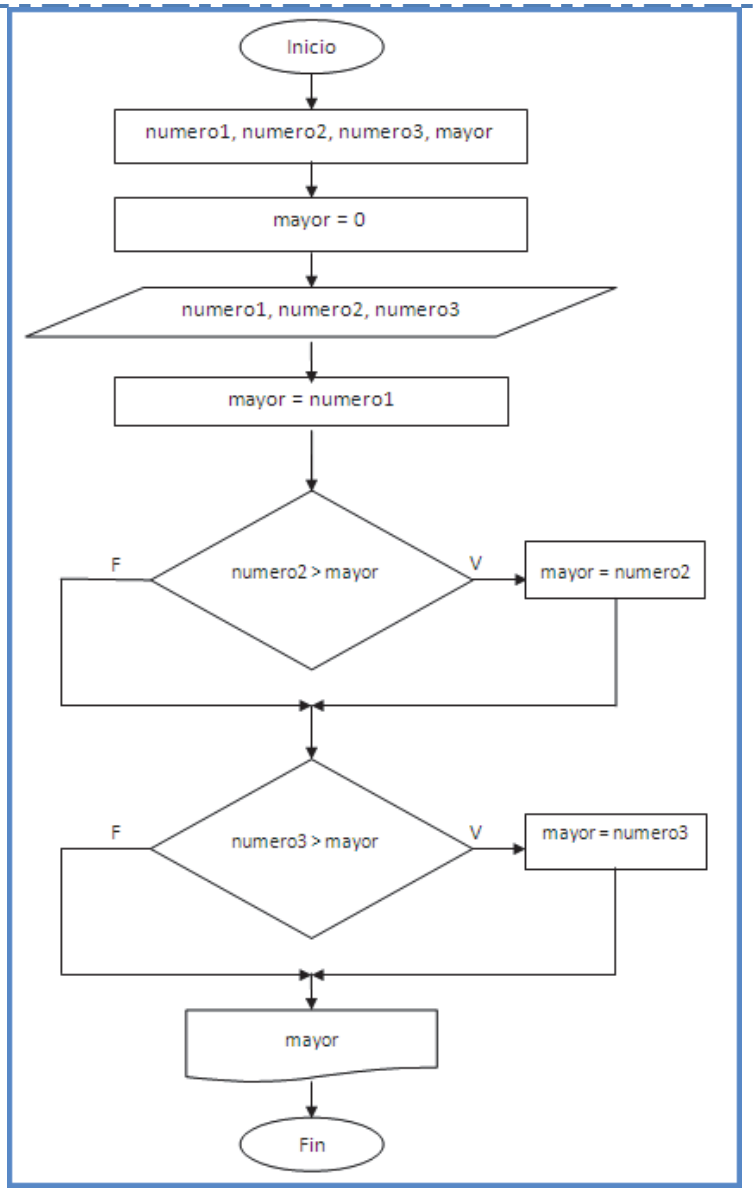
Diagrama de flujo

Implementación del programa

Ejercicio de aplicación 1

1) Elabora el diseño e implementación de un programa que permita calcular el sueldo neto de un trabajador, teniendo como datos de entrada su sueldo base y el número de hijos. Al trabajador se le descuenta el 5% de su sueldo por el seguro social, pero si tiene más de 3 hijos se le paga S/.100 soles adicionales.

2) A continuación se muestra el diagrama de flujo del diseño de un programa que permite calcular el mayor de tres números.



Se te pide hacer la implementación en VBA del programa leyendo los datos desde un formulario.

4.1.3 Condicionales anidadas

Se utiliza cuando una de las sentencias de una estructura condicional es otra estructura condicional.

Por ejemplo: en una tienda comercial, desean hacer un descuento a sus clientes tomando como primera condición que sean de sexo femenino y, dentro de esta condición, que sean mayores de 50 años.

Existen diferentes formas de anidar estructuras condicionales, por lo que no existe una sintaxis única. Aquí un ejemplo de sintaxis basado en el ejemplo anterior:

```
If sexo=femenino then
        If edad>50 then
                Accede al descuento
        Else
                No accede al descuento
        EndIf
Else
        No accede al descuento
EndIf
```

Cuando trabajas con estructuras condicionales anidadas, todo bloque de sentencias que tenga más de una instrucción debe terminar con **End If**

Hazlo tú mismo

Elabora el diagrama de flujo del pseudocódigo mostrado anteriormente.

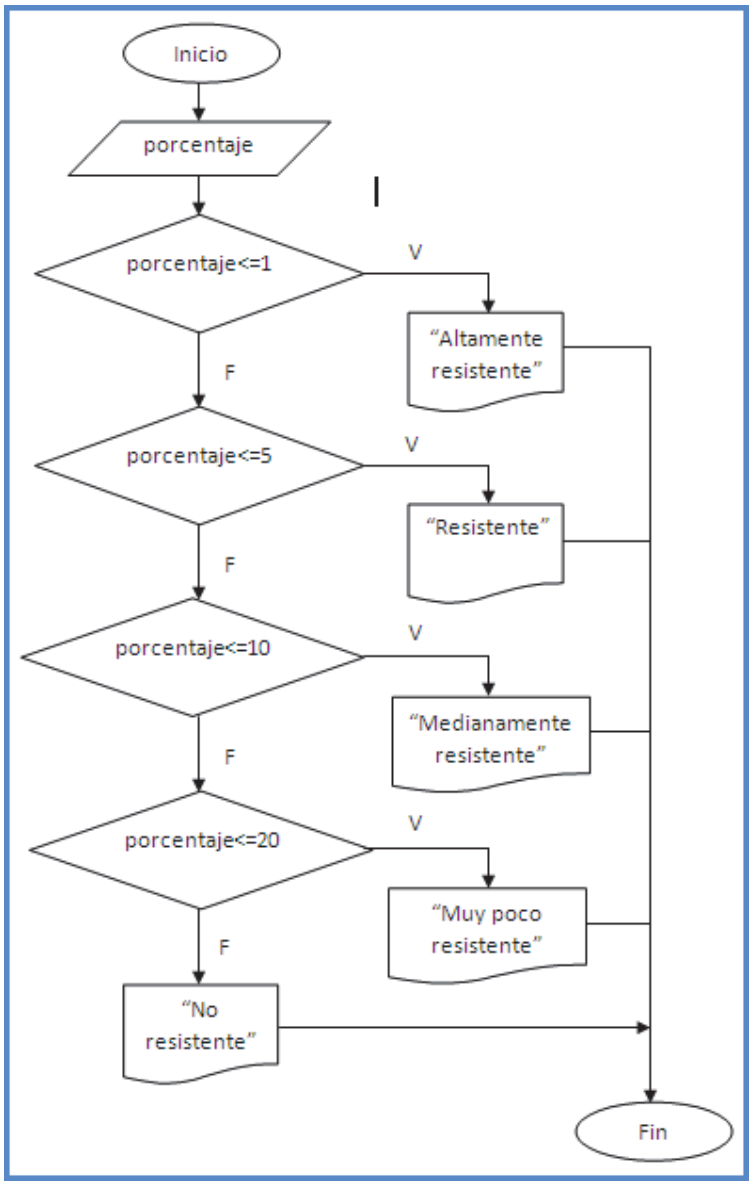
Observa el siguiente ejemplo de aplicación:

Se desea elaborar un programa en VBA que permita clasificar la resistencia de un producto cuando se lo expone a altas temperaturas, teniendo como dato de entrada el porcentaje de pérdida de peso del producto y como salida la clasificación correspondiente.

Porcentaje de pérdida de peso	Clasificación
Menor o igual a 1	Altamente resistente
Mayor a 1 pero menor o igual a 5	Resistente

Mayor a 5 pero menor o igual a 10	Medianamente resistente
Mayor a 10 pero menor o igual a 20	Muy poco resistente
Mayor a 20	No resistente

Diagrama de flujo



Implementación del programa

```
Sub Programa()
    Dim pp As Single

    pp = InputBox("Ingrese el porcentaje de pérdida de peso")
    If (pp <= 1) Then
        MsgBox ("Altamente resistente")
    Else
        If (pp <= 5) Then
            MsgBox ("Resistente")
        Else
            If (pp <= 10) Then
                MsgBox ("Medianamente resistente")
            Else
                If (pp <= 20) Then
                    MsgBox ("Muy poco resistente")
                Else
                    MsgBox ("No resistente")
                End If
            End If
        End If
    End If

End Sub
```

Hazlo tú mismo

Elabora y prueba el programa mostrado anteriormente utilizando VBA.

4.1.4 Condicional múltiple de la forma *Select…case*

Permite ejecutar un conjunto de sentencias que posee varios bloques, similar a la sentencia If...then...else, con la diferencia de que el código es más comprensible al existir varias opciones.

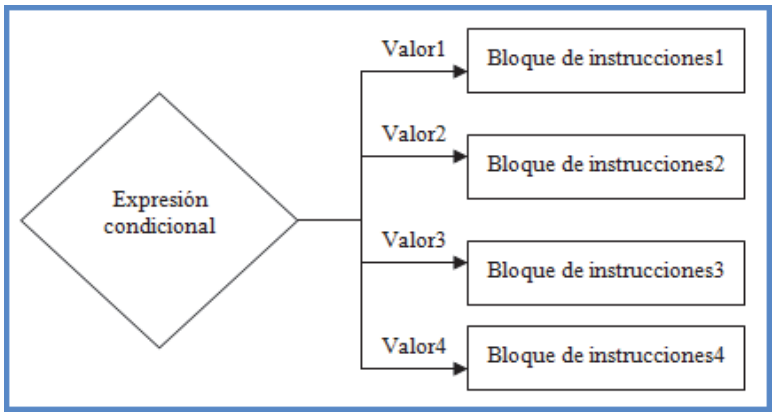
Por ejemplo: se desea elaborar un programa que reciba como dato de entrada un número entero entre 1 y 7, y luego escriba el día de la semana correspondiente. Es decir: 1=Lunes, 2=Martes, 3=Miércoles, 4=Jueves, 5=Viernes, 6=Sábado, 7=Domingo

La sintaxis de una condicional múltiple es la siguiente:

```
Select Case <expresión>
        Case lista 1
                <sentencias 1>
        Case lista 2
                <sentencias 2>
        Case lista 3
                <sentencias 3>
        .
        .
        .
        Case Else
                <sentencias n>
End Select
```

Aquí se ejecutan las sentencias ubicadas en la lista que coincidan o contengan el valor de la expresión.

Si la expresión no se encuentra en ninguna de las listas, entonces se ejecutan las sentencias del bloque **Case Else**. Este es opcional, se coloca solo si es necesario.
Su diagrama de flujo es el siguiente:



Hazlo tú mismo

Teniendo como referencia el ejemplo mencionado anteriormente (imprimir el día de la semana basándote en el número ingresado como dato), elabora el diagrama de flujo que te permita resolver el problema.

Implementación del programa

```
Sub Programa()
    Dim num As Integer

    num = InputBox("Escribe un número entre 1 y 7")

    Select Case num
        Case 1
            MsgBox ("Lunes")
        Case 2
            MsgBox ("Martes")
        Case 3
            MgBox ("Miércoles")
        Case 4
            MsgBox ("Jueves")
        Case 5
            MsgBox ("Viernes")
        Case 6
            MsgBox ("Sábado")
        Case 7
            MsgBox ("Domingo")
        Case Else
            MsgBox ("Número fuera de rango")
    End Select
End Sub
```

A continuación, observa y completa el siguiente ejemplo de aplicación sobre condicionales múltiples:

Una cadena de cines, cuya entrada es de 10 soles por persona, presenta la siguiente oferta a sus clientes:

- Si compra 1 entrada, descuento =0
- Si compra 2 entradas, descuento =0.1%*importe
- Si compra 3 o 4 entradas, descuento =0.2%*importe
- Si compra 5 o 20 entradas, descuento =0.3%*importe

Se pide elaborar un programa que permita calcular el descuento y el importe neto a pagar, utilizando el siguiente formulario.

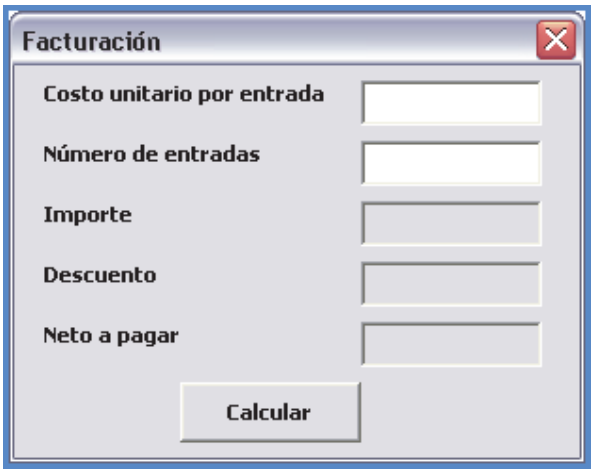
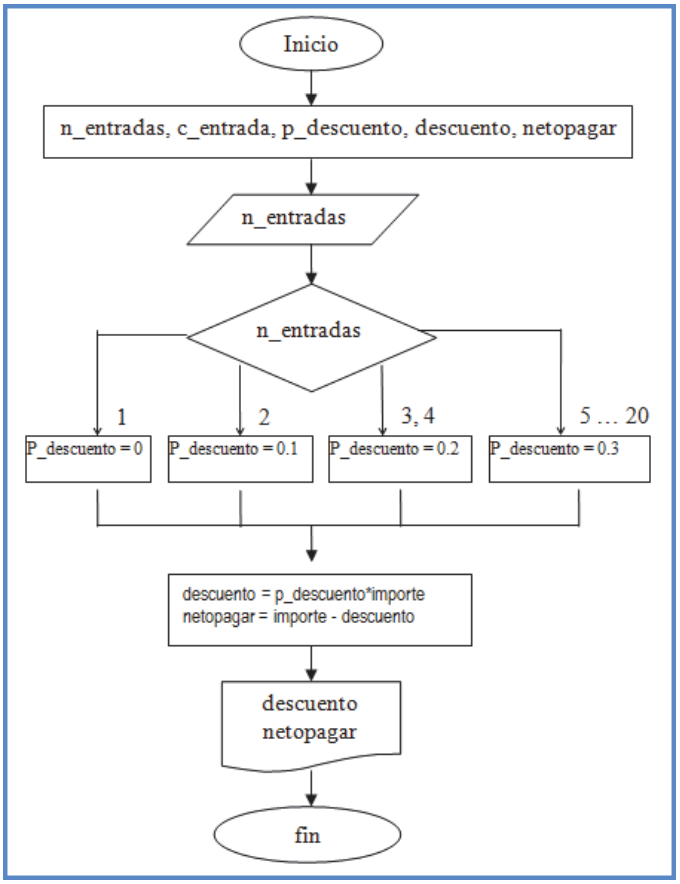


Diagrama de flujo



Hazlo tú mismo

Elabora la implementación del programa.

Ahora observa el siguiente ejemplo de condicionales múltiples utilizando el control ListBox.

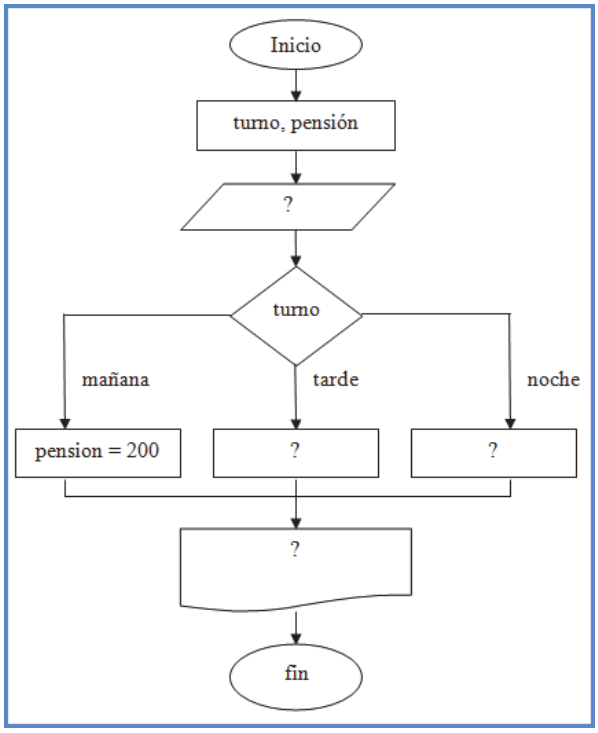
Un instituto cobra su pensión a los estudiantes según el turno al que se matriculan.

- Para el turno mañana, la pensión es 200.
- Para el turno tarde, la pensión es 220.
- Para el turno noche, la pensión es 180.

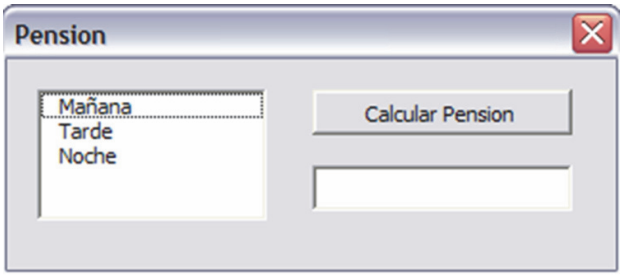
Se pide cargar en un ListBox los turnos y, según el turno elegido, mostrar en un control TextBox el valor de la pensión que se debería pagar.

Hazlo tú mismo

Completa el diagrama de flujo.



Diseño del formulario



Implementación del programa

El control **ListBox** permite mostrar una lista de elementos. Los índices de la lista ListBox turno inician en cero.

Turno	Índice
Mañana	0
Tarde	1
Noche	2

ListBox1.ListIndex devuelve el índice del elemento elegido.

```
Private Sub UserForm_Activate()
  ListBox1.AddItem ("Mañana")
  ListBox1.AddItem ("Tarde")
  ListBox1.AddItem ("Noche")
End Sub

Private Sub CommandButton1_Click()
    Select Case ListBox1.ListIndex
          Case 0: TextBox1.Text = 200
          Case 1: TextBox1.Text = 220
          Case 2: TextBox1.Text = 180
    End Select
End Sub
```

Ejercicio de aplicación 2

1) Elabora el diseño e implementación de un programa que permita clasificar a una persona de acuerdo a su edad basándote en la siguiente tabla:

Edad	Clasificación
De 0 a 1 año	Bebé
De 2 a 12 años	Niño
De 13 a 18 años	Adolescente
De 19 a 69 años	Adulto
Más de 70 años	Anciano

La edad debe ser ingresada por el usuario.

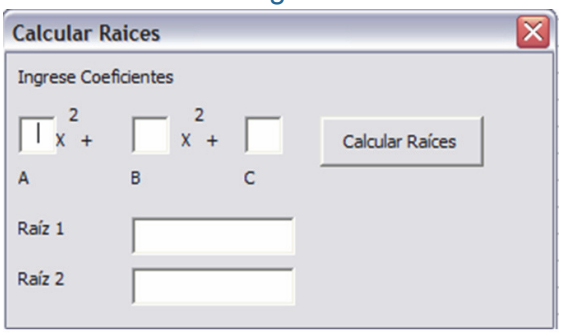
2) Se desea elaborar un programa que permita calcular las raíces de una ecuación de segundo grado ($Ax^2 + Bx + C = 0$).

Recuerda que la fórmula para el cálculo de raíces de una ecuación de segundo grado es la siguiente: $\frac{-B \pm \sqrt{B^2 - 4AC}}{2A}$

Se te pide elaborar el programa utilizando el siguiente pseudocódigo:

```
'Declarar variables
imaginario de tipo lógico
discriminante, raíz1, raíz2 de tipo real

ingresar coeficientes A, B y C

'Asumir que no tiene raíces imaginarias
imaginario = falso

Si A=0 y B=0, entonces
        Imprimir "Ecuación sin raíces"
Caso contrario
        Si A=0 'por teoría
                Imprimir "No es ecuación de segundo grado"
        Caso contrario
                'Calcular discriminante
                Discriminante = B² – 4*A*C
                Si discriminante < 0, entonces
                        'por teoría las raíces son imaginarias
                        Imprimir "las raíces son imaginarias"
                Caso contrario
                        'Cálculo de las raíces
                        raíz1 = (-B + (Discriminante)^(1/2)) / (2*A)
                        raíz2 = (-B - (Discriminante)^(1/2)) / (2*A)
                Fin del "Si"
        Fin del "Si"
Fin del "Si"
```

El formulario que debes usar es el siguiente:



3) Abre el archivo de tu trabajo de este capítulo, ingresa al editor de VBA y crea una función llamada **Avance,** que reciba como parámetros el nombre del corredor y un número real entre 0 y 1. Esta función debe devolver un valor entero que represente el avance de la liebre o la tortuga (según el parámetro ingresado) de acuerdo a las siguientes condiciones:
   a) Avance de la liebre: si el número ingresado como parámetro es menor o igual a 0.75, entonces el avance es igual a 1. Si es menor o igual a 0.9, el avance es 2. En otro caso, el avance es igual a 3.
   b) Avance de la tortuga: si el número ingresado como parámetro es menor o igual a 0.3, entonces el avance es 1, en caso contrario, el avance es igual a 0.

4.2 Estructuras de control de ciclos repetitivos

Las estructuras de repetición permiten ejecutar una o más líneas de código repetidamente. A los bloques de instrucciones repetitivos, se les conoce como **bucles o iteraciones.**

Por ejemplo:

Si se desea elaborar un programa que imprima los 100 primeros números naturales, con lo que conoces hasta el momento lo harías de la siguiente manera.

```
Imprimir 1
Imprimir 2
Imprimir 3
...
Imprimir 100
```

← estructura secuencial

Pero utilizando estructuras de control de ciclos repetitivos, lo podrías hacer con un algoritmo más simple como este.

```
Para numero=1 hasta 100
    Imprimir numero
Fin Para
```

← estructura iterativa

En VBA, los principales tipos de estructuras de control de ciclos repetitivos son los siguientes:

- Ciclos de la forma: **For.... Next**
- Ciclos de la forma: **Do…Loop While**

4.2.1 Ciclos de la forma *For…Next*

Se utiliza cuando se conoce el número exacto de repeticiones que se van a ejecutar.

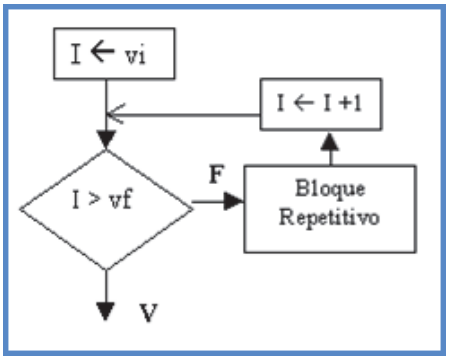
Su sintaxis es la siguiente:

```
For contador=<valor inicial> To <valor final> Step <incremento>
        <sentencias>
Next
```

El bloque de sentencias es ejecutado tantas veces como lo indique el contador desde su valor inicial hasta su valor final, tomando en cuenta el incremento o decremento indicado en **Step**.

El **contador** es una forma de controlar las iteraciones realizadas. Este sufre un incremento o decremento en una cantidad fija en cada iteración. Si no se coloca **Step**, el contador se incrementará en 1 en cada iteración.

Su diagrama de flujo es el siguiente:



Donde:
I: es el contador
vi: es el valor inicial del contador
vf: es el valor final del contador

Observa el siguiente ejemplo de aplicación:

Se desea elaborar un programa que permita imprimir los números pares del 2 al 10 (ambos inclusive).

```
Sub pares()
    Dim i As Integer

    For i = 2 To 10 Step 2
        MsgBox (i)
    Next
End Sub
```

Un **acumulador o totalizador** tiene como objetivo guardar las sumas sucesivas de una variable resultante dentro de una iteración.

Por ejemplo: si la variable *nuevo_valor* se va calculando dentro de una iteración, entonces si deseas acumular los valores de esta variable, debes hacer lo siguiente dentro de la iteración:

suma =suma+nuevo_valor

donde *suma* es el acumulador.

A continuación un ejemplo de aplicación con el uso de un acumulador:

Se desea elaborar un programa que permita calcular el valor de la siguiente serie:

$$1+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+...+\frac{1}{n}$$

Donde "n" es un número ingresado por el usuario.

```
Sub serie()
    Dim n As Integer, i As Integer
    Dim suma As Single

    n = InputBox("Ingrese el valor de n")
    suma = 0 'Se inicializa el acumulador

    For i = 1 To n 'El incremento será 1
        suma = suma + 1 / i
    Next

    MsgBox ("suma: " & suma)
End Sub
```

Hazlo tú mismo

1) Ingresa al editor de VBA, elabora y prueba los dos programas anteriores.
2) Modifica el programa que calcula los números pares para que devuelva los números impares del 1 al 15.

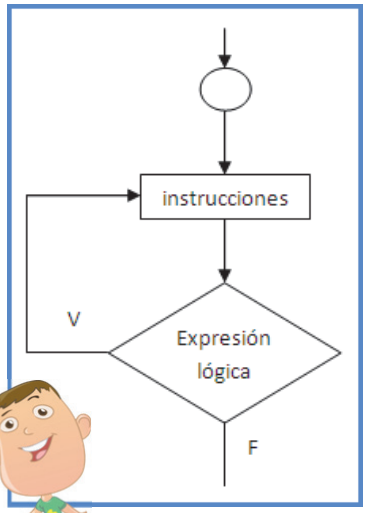
4.2.2 Ciclos de la forma *Do….Loop While*

Este tipo de estructura se usa cuando no se conoce el número de iteraciones. Las instrucciones se ejecutarán mientras la condición establecida sea verdadera.

Su sintaxis es la siguiente:

```
Do
        <sentencias>
Loop While <expresión lógica>
```

Su diagrama de flujo es el siguiente:



Un **interruptor** (*switch*, centinela, bandera o *flag*) te permiten controlar si se ejecutarán ciertos bloques de programa. Tiene dos valores diferentes a lo largo de la ejecución del programa, estos pueden ser 1 y 0, verdadero y falso, sí y no, etcétera.

Observa el siguiente ejemplo de aplicación:

Se desea elaborar un programa que permita calcular el promedio de notas del curso de computación. El programa se detendrá cuando el usuario ingrese un valor negativo.

Implementación del programa

```
Sub promedio()
    Dim nota As Single, suma As Single, promedio As Single
    Dim i As Integer
    Dim salir As Boolean

    suma = 0
    i = 0
    salir = False

    Do
        nota = InputBox("Ingrese nota")
        If (nota > 0) Then
            suma = suma + nota
            i = i + 1
        Else
            salir = True
        End If
    Loop While (salir = False)

    promedio = suma / i
    MsgBox ("El promedio es : " & promedio)
End Sub
```

Analiza el código del programa anterior y responde:

- ¿Cuál es el nombre de la variable que funciona como “contador”?
  _____
- ¿Cuál es el nombre de la variable que funciona como “acumulador”?
  _____
- ¿Cuál es el nombre de la variable que funciona como “interruptor”?
  _____
- ¿Cuántas iteraciones hace el programa? Comenta_____
  _____

Hazlo tú mismo

Ingresa al editor de VBA, elabora y prueba el programa anterior.

Dependiendo del problema presentado, en ocasiones es conveniente colocar la expresión lógica al inicio del bucle, por lo que su sintaxis tomaría la siguiente forma:

Do While (expresión lógica)

<sentencias>

Loop

Ahora observa y completa el siguiente programa:

El profesor de computación desea elaborar un programa que permita obtener información acerca de los alumnos de un aula.

- Número de alumnos de sexo masculino
- Número de alumnos de sexo femenino
- Edad promedio
- Nota promedio del aula

El programa leerá como primer dato el número de alumnos del aula y, luego, utilizando una estructura de repetición, para cada alumno se debe leer el sexo, la edad y su nota.

Completa la implementación del programa. Las instrucciones faltantes están como comentarios.

```
Sub Programa()
    Dim num As Integer, sexo As String, edad As Byte, nota As Single
    Dim i As Integer, masc As Integer, feme As Integer
    Dim sumaedad As Single, sumanota As Single, edadpr As Single, notapr As Single

    'en esta línea debes leer el número de alumnos y asignarlo a la sariable "num"

    'en esta línea inicializa el contador i

    'en esta línea inicializa el contador masc

    'en esta línea inicializa el contador feme

    Do While (i < num)
        sexo = InputBox("Sexo: ", "Alumno " & i)
        edad = InputBox("Edad: ", "Alumno " & i)
        'en esta línea lee la nota y asígnalo a la variable "nota"

        If (sexo = "M" Or sexo = "m") Then
            'en esta línea incrementa el contador masc en 1

        Else
            If (sexo = "F" Or sexo = "f") Then
                feme = feme + 1
            End If
        End If
        'en esta línea incrementa al acumulador sumaedad la edad leída

        sumanota = sumanota + nota
        i = i + 1
    Loop
    edadpr = sumaedad / num
    'en esta línea debes calcular la nota promedio y asignarlo a la variable "notapr"

    MsgBox ("Edad promedio: " & edadpr)
    MsgBox ("Nota promedio: " & notapr)
    MsgBox ("Nro. de hombres: " & masc)
    'en esta línea muestra el número de mujeres


End Sub
```

Ejercicio de aplicación 3

1) Diseña e implementa un programa que permita calcular el factorial de un número. Recuerda que el factorial de un número se halla de la siguiente manera:

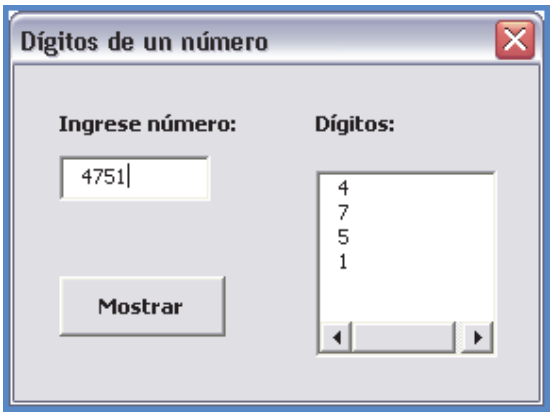
Factorial (N)=1*2*3*…..*N

Se sugiere que sigas estos pasos:

a) Crea una función factorial que reciba como parámetro un número y que aplique la fórmula indicada anteriormente para calcular el factorial de dicho número.

b) Crea un programa principal, que lea un número y, si es mayor que cero, calcule su factorial, en caso contrario debe mostrar el mensaje "El número debe ser mayor que cero".

2) Diseña y elabora un programa que lea un número y muestre cada uno de sus dígitos en un control ListBox, tal como se muestra en la figura:



3) Abre tu archivo **simulación.xlsm**, ingresa al editor de VBA y haz lo siguiente:

a) Crea un subprograma **Simula** que reciba como parámetro la distancia a recorrer en la carrera y devuelve el nombre del ganador de la carrera. Por cada iteración, el subprograma deberá determinar el avance de cada corredor, para ello deberás invocar a la función **Avance** (que ya la creaste anteriormente). Para generar un número aleatorio entre 0 y 1, debes utilizar la función Rnd().

b) Crea un subprograma **Principal**, que lea la distancia de la carrera y el número de simulaciones a realizar. Luego, que simule N veces la carrera, cuente cuántas veces ganó cada corredor y muestre los resultados.

c) Prueba la ejecución de tu programa.

¿CUÁNTO APRENDÍ?

I. Contesta las siguientes preguntas:

a) ¿Cuándo se utilizan las estructuras de control repetitivas?

_____

_____

b) ¿Cuál es la diferencia entre un acumulador y un contador? Escribe un ejemplo para cada uno de ellos.

_____

_____

c) Analiza las siguientes instrucciones>

```
Dim i as Integer
For i=0 To 68  Step 2
        <sentencias o instrucciones>
Next
```

¿Cuántas iteraciones se ejecutarán en el bucle mostrado anteriormente?_____

d) Analiza las siguientes instrucciones:

```
Dim i as Integer, suma as Integer
suma=0
For i=3 To 10
        suma=suma+i
Next
```

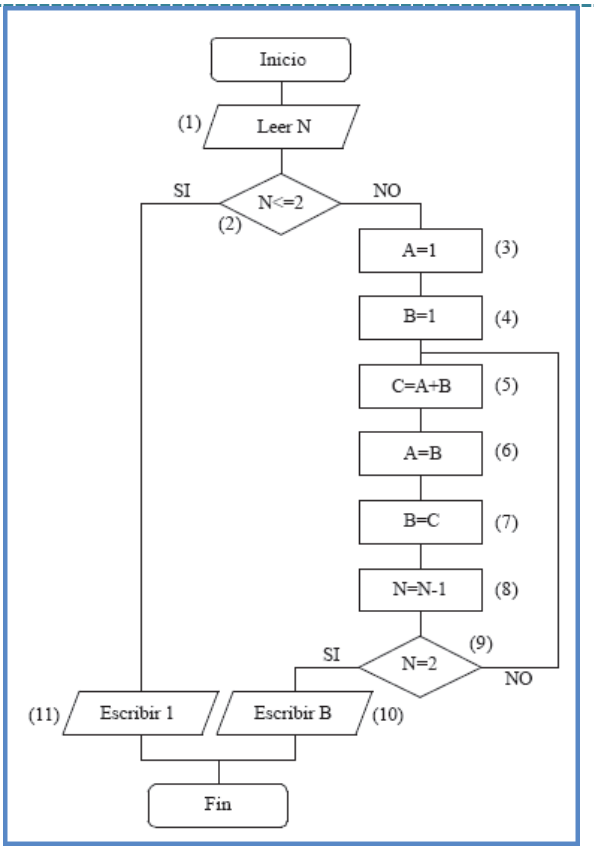
¿Cuál es el valor final de la variable **"suma"**?_____

II. Elabora el diseño e implementación de un programa que calcule el promedio de 4 evaluaciones de un estudiante. Si el promedio es mayor a 10.5, debe mostrar el mensaje "Aprobado", en caso contrario, debe mostrar el mensaje "Desaprobado".

III. La sucesión de Fibonacci se define así: $a_1=1$, $a_2=1$, $a_n=a_{n-1}+a_{n-2}$ para $n>2$

Por ejemplo, estos son los primeros términos de la sucesión: 1, 1, 2, 3, 5, 8….

Elabora un programa que permita calcular cualquier término de la sucesión deFibonacci, tomando en cuenta el siguiente diagrama de flujo:



IV. Diseña e implementa un programa que permita evaluar la siguiente función dado un valor de X:

$$Y = \begin{cases} 3X & \text{si} \quad X < 10 \\ X^2 - 3 & \text{si} \quad 10 \leq X < 20 \\ \sqrt{X} & \text{si} \quad X \geq 20 \end{cases}$$

Cuarta etapa

En esta última etapa culminarás el desarrollo de tu calculadora con güincha, para ello, debes realizar lo siguiente:

1. Abre el archivo de tu proyecto.
2. Implementa las líneas de código necesarias para que se puedan almacenar las operaciones que vas realizando.
3. Averigua e implementa cómo exportar las operaciones a una hoja de Excel.